Universidad Autónoma de Madrid

"Diseño y resolución interactiva de ejercicios que involucren cálculo simbólico"

TESIS DOCTORAL

Autor: **Fernando Díez Rubio**

Director: **Roberto Moriyón Salomón**

Departamento de Ingeniería Informática

Escuela Técnica Superior de Informática

Noviembre de 2001

Memoria presentada para optar al título de Doctor en Ingeniería Informática

A Carmen,

"...Ahora siento que llegó el día
que tengo ganas de vivir,
de atravesar los muros y ruinas
que aunque pase el tiempo están ahí
y florecer como un hombre nuevo
sin miedo a las tragedias por venir.

Regalarle a la vida todo el fuego
de tus ojos y tus ansias de vivir.

Iba vestida la aurora
con rayos de sol
y en los cabellos prendida
llevaba una flor".

Triana

Agradecimientos

Vaya mi primer agradecimiento para ti Roberto. Has sido el director de este trabajo. Pero, por encima de la figura administrativa está la de tu amistad. Durante los años que ha durado la investigación no has tenido el menor inconveniente en dedicarme las horas más intempestivas que hemos encontrado en común para trabajar: fines de semana, madrugones, días sin comer o comiendo a toda prisa por podernos reunir un rato. Y siempre, daba igual el momento que fuera, me has mostrado tu lado más amable y tu paciente dedicación. A pesar de los palos que me arreaste al principio desengañándome del trabajo que yo quería hacer y haciéndome ver que primero había que avanzar en lo que hoy es *MathEdu*, a cambio siempre me ofreciste tu ayuda y dedicación. Trabajando contigo he disfrutado de momentos inolvidables y por todo ello te estaré siempre agradecido.

La historia de esta memoria es ya veterana. Mis circunstancias personales, laborales y familiares, han dilatado en el tiempo un trabajo que, en otras circunstancias tal vez debería haber durado menos tiempo. Durante todo este periodo he tratado con múltiples personas que han influido en el resultado final. Voy a tratar de recordarlas aquí para expresarles mi agradecimiento por su apoyo. Probablemente se me olvidará alguna pero, si alguien no se llega a encontrar reflejado aquí, que no me guarde rencor. La culpa es exclusivamente mía.

Y debo comenzar por mi amigo Pablo Castells. Gracias a aquellas largas charlas que manteníamos cuando estuve en el IIC empecé a imponerle ciertas condiciones a mi destino. Hoy puedo decir que algunas ya las he conseguido, entre otras esta. Gracias por darme ánimos cuando he flojeado y por tu optimista y contagiosa forma de ser. Por cierto, ¿cuándo corremos?.

Ha habido otras personas con las que la relación ha sido más efímera pero muy intensa. Con ellas he compartido tal vez los momentos más amargos de este trabajo pero, a la vez, de creatividad más arrolladora. Sois mis amigos de la Nebrija (hoy ya casi todos "ex"): Ainhoa, Almudena, Antonio, Beatriz, Carmen mi paciente amiga, que me padeciste como compañero de mesa sin mandarme a hacer puñetas por no hacerte ni caso en tantas ocasiones, Cova, David (Sr. UNNE), Edu, Eloy, Isabel, Jaime, Joze Lui, Juan Abel, Excelentísimo y Magnífico Manuel, Manuel Gómez (ojalá ya seas doctor), María José, Mariel, Mercedes, Olga, Pilar, Robertón, Rosa, Tresca, Yolanda y seguro que me dejo a alguien. Muchas gracias y mucha suerte a todos.

A muchos de los que actualmente sois mis compañeros en la ETSI os conocí en el IIC hace un buen montón de años. A otros os acabo de conocer. En aquel entonces me contagiasteis de un importante espíritu de compañerismo, y siempre que he vuelto por la UAM y me he encontrado con cualquiera de vosotros por los pasillos me habéis saludado

como a uno más, como si todo este tiempo hubiera estado aquí. Gracias a Alejandro, Ana, Carlos Santa Cruz, Estrella, José Antonio, José Dorronsoro, Juan Alberto, Manuel Alfonseca, Miguel Ángel, Pablo Varona, Paco Borja, Paco Sáiz, Pilar, Ramón, Rosa, Ruth, Xavier y a todos aquellos que me habéis estimulado y acogido tan bien. Un recuerdo especial a Julio Gonzalo que en todo este tiempo también me ha transmitido su apoyo.

El desarrollo de esta Tesis ha coincidido en el tiempo con la edificación de esta magnífica Escuela en la que nos encontramos. Agradezco a Javier Garrido por confiar en mi trabajo y darme la oportunidad pertenecer a la misma. Gracias, Juana, por tu inestimable ayuda, eficacia en estado puro. Y, por supuesto, a Angel, Esther, Eugenio y todos los que hacéis posible que la Escuela sea un lugar envidiable para trabajar.

Un agradecimiento especial, recompensable en forma de *Olla Podrida*, para mis hermanos Ángel y Alfonso y mi cuñada Carmen por vuestras revisiones y los comentarios de última hora. Y a todos, cuñados, suegros, sobrinos, gracias por vuestro interés y por la ayuda para evadirme de las preocupaciones. Por cierto, chicas, ¡preparaos para perder el siguiente Parchís!. También me ayudaron a evadirme periódicamente mis gastronómicos amigos *Cofrades* y así os lo reconozco: ¡os merecéis todos un buen *pil-pil*!. Y también un recuerdo para mi padre, que vivió con ilusión los días en que empezaba este trabajo y supo transmitirme confianza en que algún día lo terminaría con éxito. Aquí está.

Y me faltan las personas claves en esta historia. Al comenzar los cursos de doctorado sólo éramos dos y hoy ya somos cinco en casa. Con tanto bullicio las cosas no han sido fáciles, pero en los momentos difíciles siempre he encontrado al otro lado de la puerta del estudio cariño, comprensión, arrebatadoras sonrisas de los niños y mogollón de pañales que cambiar (que también ayudan a evadirse). Así es que para acabar quiero hacer una dedicatoria muy especial a mis hijos y a mi mujer. A **Irene**, por obligarme cada día a madrugar un poquito más. Sin ti probablemente no habría tenido la fuerza de voluntad ni el tiempo suficiente. A **Elena**, porque en tu carácter se refleja toda la fuerza y la alegría de vivir, tan contagiosa, "¿a que sí?". **Alejandro**, el peque. Tu añito de vida se solapa con esta memoria pues habéis nacido al mismo tiempo. Has sido el estímulo perfecto para dar el empujón final y echar a andar.

Y a ti **Carmen**, mi paciente y dedicada *María*, te dedico todo este trabajo. Gracias de todo corazón.

Tabla de contenidos

Índice de ilustraciones

Índice de tablas y figuras

Tablas

Figuras

Resumen

La aparición del PC a principios de los años 80 ha supuesto una auténtica revolución en nuestra forma de vida y en nuestros comportamientos sociales actuales. No hace aún un cuarto de siglo desde que IBM lanzó al mercado su *Personal Computer* y hoy en día un gran número de nuestras actividades cotidianas están, de un modo u otro, regidas, controladas o supervisadas por un ordenador y sus programas. Es altamente probable que dentro de un siglo los libros de texto escolares hablen de la Revolución Tecnológica que hoy día estamos teniendo la suerte de vivir y de la que somos auténticos protagonistas. Pero habitualmente se tiende a olvidar que los ordenadores, como tales, no son más que meros artefactos electrónicos, altamente sofisticados, pero carentes de capacidad alguna. Toda su potencia, su capacidad de cálculo, de control y supervisión de procesos o tareas, está ineludiblemente ligada a los programas que los controlan.

Este ambiente de euforia informática en que nos desenvolvemos a diario no debería ensombrecer ciertas actividades cotidianas en las que los avances logrados son más lentos o, por decirlo de otro modo, menos llamativos. Gracias a Internet es posible realizar actividades tan diversas como operaciones bancarias, hacer la compra, encargar unas flores, leer la prensa o sacar entradas para el teatro. Pero si pensamos en la informática educativa, en la actividad personal del alumno frente al ordenador, empezamos a encontrar espacios aún sin llenar. Qué duda cabe de que existen herramientas altamente sofisticadas como sistemas de diseño asistido por ordenador, simuladores, sistemas de cálculo simbólico, etc. Pero si se trata de que el ordenador no sólo responda a los estímulos, tratados como parámetros, que le llegan del alumno, sino que los interprete y sea capaz de provocar en este razonamientos, deducciones o cálculos complejos, la dificultad en los avances de la informática se incrementa notablemente.

En este contexto de informática educativa personal, un tanto olvidado en la actualidad en beneficio de la globalización de la educación, hemos querido aportar con *MathEdu* un pequeño grano de arena. *MathEdu* representa una aproximación a la interacción persona-ordenador a través de una aportación metodológica básica para el desarrollo de cursos que involucren cálculo simbólico mediante una herramienta de autor. *MathEdu* incluye además dos aplicaciones para la resolución, interactiva o guiada, de los ejercicios modelados en el curso. *MathEdu* no incluye aspectos que hoy en día priman, fundamentalmente los avances en sistemas que involucren colaboración, adaptación y accesibilidad a través de Internet. Sin embargo pensamos que la línea en la que avanzamos necesariamente es transcendente en el sentido de que cada vez van a ser mayores las necesidades de una comunicación profunda en la interacción persona-ordenador, en todos los aspectos relacionados con la Ciencia y, en particular, en el simbólico. Este es el motivo que nos anima a seguir en esta investigación.

MathEdu representa un paso pequeño pero necesario hacia la consecución del paradigma de interacción que comentamos. Para desarrollar las componentes de *MathEdu* nos hemos basado en *Mathematica*, que sin lugar a dudas es uno de los mejores programas de cálculo simbólico. Con *Mathematica* podemos trabajar en el plano simbólico, y tratamos de conseguir en el alumno el efecto de que su interlocutor es un profesor que es capaz de interpretar las respuestas que este da a las preguntas que le plantea, así como plantearle nuevas preguntas relacionadas con las respuestas anteriores. Para conseguir este efecto hemos tenido que desarrollar un modelo que sea capaz de albergar los datos contenidos en cualquier ejercicio de Matemáticas que involucre cálculo simbólico (derivadas, integrales, límites, etc.). Con esto no estamos diciendo que el sistema sea capaz de interpretar el Lenguaje Natural, pues de hecho no es lo que pretendemos. Por el momento nos restringimos, que no es poco, a la interpretación de una parte del lenguaje simbólico matemático.

La memoria que presentamos consta de tres partes. En la primera de ellas hacemos una introducción a la problemática que tratamos de resolver y situamos el contexto tecnológico en el que se desenvuelve nuestra investigación.

La segunda parte está dedicada íntegramente a *MathEdu*. En ella exponemos desde la arquitectura global del sistema hasta las componentes esenciales del mismo, incidiendo continuamente en las estructuras de datos que se manejan y en las interfaces desarrolladas.

Finalmente, en la tercera parte se exponen las pruebas efectuadas con el sistema y las conclusiones derivadas de las mismas, así como las posibilidades de futuras investigaciones asociadas o derivadas de la que a continuación presentamos.

Es nuestro deseo que el lector encuentre en las páginas que siguen motivos suficientes de interés por el trabajo desarrollado. Hemos procurado que la memoria sea conceptualmente clara y afable su lectura, procurando no ahondar más de lo estrictamente necesario en detalles excesivamente técnicos o abstractos que pudieran desviar la atención de las ideas más relevantes. Esperamos haberlo logrado.

El autor

Abstract

The appearance of the PC at the beginning of the 80's has supposed actually a revolution in our lifes and our social behaviours. Less than a quarter of a century has passed since IBM's Personal Computer arrived to the market and nowadays a great majority of our daily activities are, in one way or another, governed, controlled or supervised by a computer and its programs. It is highly probable that in one century the school textbooks will speak about the Technological Revolution that we are living today. But usually it is forgotten that the computers are nothing but electronic machines, highly sophisticated, but lacking in any capacity. All their power and their capacity of calculation, control and supervision of processes or tasks, is unavoidably tied to the programs that control them.

This environment of euphoria about computers in which we are involved daily should not shade us the existence of certain activities in which the successful advances are slower or less showy. Thanks to Internet it is possible to accomplish activities as diverse as bank operations, shopping, ordering flowers, reading the press or buying tickets for the theatre. But if we think about educational computer science, about the personal activity of the student in front of the computer, we start to find unexplored spaces. There are no doubts that today there are highly sophisticated tools for education as computer aided design systems, simulators, systems of symbolic calculation, etc. But if the objective is that the computer not only answers to the stimuli which come from the student treated as parameters, but we also want that the computer interprets them and it encourages the student's reasoning, making deductions or complex calculations, the difficulty in the technological advances that are needed is increased notably.

In this context of personal educational computer science, slightly forgotten at the present time in benefit of the globalisation of the education, our goal has been to make a contribution that can be useful as a stone in a path to be built. *MathEdu* represents an approximation to the human-computer interaction through a methodological basic contribution for the development of courses that involve symbolic calculation by means of an author tool. It includes two applications for the resolution, interactive or guided, of the exercises modelled in the course. *MathEdu* does not include aspects that today has the main priority given to the advances in systems that involve collaboration, adaptability and accessibility through Internet. Nevertheless we think that the direction in which we are walking is necessarily transcendent in the sense that, the need of a deep communication during the human-computer interaction process in all the aspects related to Science, and in particular in the symbolic one, is going to be bigger and bigger each time. This encourages us to continue with this research.

MathEdu represents a small but necessary step towards the attainment of the paradigm of interaction mentioned above. In order to develop the components of *MathEdu* our work has been based on *Mathematica*, without any doubts one of the best programs for symbolic calculations. With *Mathematica* we can work at the symbolic level, trying to make the student feel as if the computer was a teacher who is capable of interpreting the answers that he gives to the questions raised, as well as to pose new questions related with the previous ones. To obtain this effect we have had to develop a model that is capable of sheltering the information contained in any exercise of mathematics that involves symbolic calculation (derivatives, integrals, limits, etc.). With this we are not saying that the system should be capable of interpreting Natural Language. For the present time we restrict our investigations to the interpretation of a section of the mathematical symbolic language.

The monography that we present consists of three parts. In the first one we do an introduction to the problematic that we try to solve and we show the technological context in which our investigation is developed.

The second part is devoted entirely to *MathEdu*. It includes a detailed explanation of the system that goes from its global architecture to the essential components of it, explaining the data structures that *MathEdu* handles and the interfaces that have been developed.

Finally, in the third part the tests carried out with the system and the conclusions derived from these are exposed, as well as the possibilities of future research associated or derived from the one that we present.

It is our desire that the reader finds in the next pages the main reasons that motivate the work that has been developed. We have tried to write a monography that is conceptually clear and with a friendly reading, trying to deepen into excessively technical or abstract details that could turn aside the attention of the most relevant ideas only when strictly necessary. We hope to have achieved it.

The author

Parte I
Introducción

Capítulo 1.

INTRODUCCIÓN

Expresó Galileo Galilei que "el libro de la naturaleza está escrito en lenguaje matemático". El empirismo, la observación de la naturaleza, basar el conocimiento en la propia experiencia eran las actitudes propias de los científicos de su época. Se recuperaba, por así decirlo, la ciencia empírica que en la Antigüedad hacía, por ejemplo, Aristóteles. La máxima a seguir era, nuevamente en palabras de Galileo "mide lo que se pueda medir, y lo que no se pueda medir, hazlo medible."

Actualmente, en los albores del nuevo milenio, se da un fiel reflejo de lo que en aquella época ocurría. El avance de la tecnología, los nuevos materiales, la informática, la Ciencia en general está entregada con frenesí al conocimiento del medio que nos circunda. La Humanidad ha roto las amarras de una ligadura secular a planteamientos panteístas de la Vida y se ha involucrado en una desenfrenada carrera hacia el Saber, el Conocimiento y el Desarrollo de la Civilización.

1.1. Origen y Objetivo del Trabajo

En este marco de avance y conocimiento del medio que vivimos, encontrábamos la carencia de un sistema que permitiera entablar un diálogo con un cierto grado de generalidad entre un sistema informático de cálculo simbólico en Matemáticas y un usuario cualquiera. Así comenzó a surgir la idea de aventurarnos en el reto de intentar *hacer medible* el conocimiento matemático, representarlo, para más adelante, poder manipularlo.

En informática, cualquier representación del conocimiento posibilita, de manera más o menos inmediata, la manipulación de objetos complejos que permiten elaborar otros objetos e informaciones también complejas basadas en aquellos. Pese a que ya existen desarrollos que realizan tareas de reconocimiento de información simbólica de conceptos matemáticos como *Calculus Wiz* (Stroyan 98), PAT (Koedinger, 1997) y otros, todos ellos presentan la carencia común del diálogo hombre-máquina. Algunos de estos sistemas, como el referido PAT, posibilitan que un usuario experimentado diseñe, de forma explícita, los enunciados de los ejercicios que más tarde se presentarán al alumno. Pero, por el contrario, no ofrecen la posibilidad de hacer ejercicios distintos de los diseñados. Otros

sistemas, sin embargo, permiten resolver distintos tipos de ejercicios cada vez que el alumno utiliza la herramienta al estar diseñados por encima de programas de cálculo simbólico de propósito general como pueden ser *Mathematica* o *Maple* (*Wiley Web Test, http://www.math.unl.edu/webtests/docs/focus2/*). En estos casos es habitual que presenten carencias en el diálogo entre el usuario y el sistema durante el proceso de resolución de los ejercicios propuestos por la dificultad que entraña saber, en cada ejecución del sistema, qué información se está presentando al usuario, al haber sido generada esta de manera aleatoria.

En definitiva, nos encontrábamos ante la circunstancia de la existencia de sistemas que utilizaban conocimiento matemático representado de diferentes formas. Todos ellos presentaban distintas posibilidades de interacción. Pero no encontrábamos alguno que permitiera elaborar procesos de representación de conocimiento matemático y, simultáneamente o *a-posteriori,* utilizara dichos procesos de representación para mostrar al alumno distintos tipos de ejercicios basados en el conocimiento elaborado y, lo que es más importante, le ayudara o le guiara en la resolución de los mismos. En tal situación el objetivo inicial del presente trabajo que nos planteamos en aquel momento fue

> Desarrollar los mecanismos necesarios para que un profesor de Matemáticas pueda diseñar los datos y procesos básicos que describan un ejercicio de cálculo. A partir de dicho diseño se pretende que los alumnos puedan resolver, de forma interactiva, cualquier otro ejercicio similar que involucre procedimientos equivalentes. Además, la aplicación resultante debe permitir la incorporación de un modelo del alumno que posibilite un aprendizaje guiado de estos.

Debe quedar claro que este objetivo de trabajo se circunscribe a métodos de cálculo modelables por el profesor. Por tanto no tratamos de enseñar a resolver problemas ni a despertar la imaginación del estudiante más allá de lo que puede hacerse en el plano de resolución algorítmica de ejercicios.

Fruto de aquel objetivo inicial de trabajo es la memoria que a continuación presentamos, la cual describe la herramienta ***MathEdu.*** Como iremos viendo en el desarrollo de los siguientes capítulos, esta herramienta posibilita dos tipos genéricos de tareas. Por una parte, permite describir los elementos necesarios para diseñar ejercicios de Matemáticas que involucren cálculo simbólico. En segundo lugar, permite resolver interactivamente cualquier ejercicio generado a partir de las especificaciones dadas durante el proceso anterior. Creemos ver cumplido, con esta primera versión de *MathEdu,* el objetivo inicial del trabajo que nos habíamos propuesto, a sabiendas de que aún es mucho el trabajo que queda por realizar como analizaremos en las conclusiones finales.

1.2. Un primer ejemplo

Con el fin de motivar una primera reflexión sobre el tipo de tareas que pueden realizarse con *MahtEdu*, vamos a exponer un primer ejemplo. Como ya hemos comentado anteriormente, *MathEdu* permite dos tipos diferentes de interacción: la del profesor (al que en adelante denominaremos genéricamente diseñador) y la del alumno (al que denominaremos, en ocasiones, usuario). Analizaremos en la primera parte del ejemplo las tareas, a grandes rasgos, que el diseñador debe realizar.

Inicialmente *MathEdu* podemos considerarlo como un recién nacido cuyo grado de conocimiento del medio que le circunda es escaso. El profesor es el encargado de definir el conocimiento en *MathEdu*. Para ello *MathEdu* está estructurado en módulos. Uno de estos módulos proporciona al profesor la capacidad de ir definiendo los contenidos necesarios para generar cursos que involucren cálculo simbólico. Pensemos entonces que el diseñador desea definir los datos y procedimientos necesarios para afrontar la resolución de integrales inmediatas de funciones trigonométricas. Por ejemplo

Ejercicio. Calcula la siguiente integral

$$\int Sen(x)dx$$

La resolución de este ejercicio es, en cierto modo, trivial. No olvidemos que para ser capaces de encontrar una primitiva de la función del integrando son necesarios, entre otros, conocimientos de trigonometría, para conocer la función que nos presentan, derivación, para ser capaces de verificar si la primitiva que propongamos como solución es la adecuada; e integración, para que a través de la teoría general de integrales indefinidas sea capaz de encontrar el conjunto de las primitivas del $Sen(x)$.

Para realizar el diseño el profesor debe aportar todos los datos necesarios que describan el ejercicio planteado a fin de que el módulo encargado de manipular dicha representación posibilite al alumno la resolución interactiva del mismo. *MathEdu* utiliza en todos su módulos una interfaz basada en *Mathematica*. A pesar de las limitaciones que ello conlleva, la manipulación de expresiones matemáticas simbólicas es altamente eficaz y, en consecuencia, hemos adoptado este modo de comunicación con el sistema. La descripción abstracta de los datos de un ejercicio se realiza mediante tres pasos realizados de forma secuencial. Así, en primer lugar, se debe definir el ejercicio a resolver, especificando cómo es el enunciado. En segundo lugar se deberán especificar las generalizaciones posibles para dicho ejercicio. Ya describiremos más adelante con detalle en qué consiste el proceso de generalización. Baste ahora indicar que este proceso corresponde a la acción de indicar los elementos del ejercicio que son susceptibles de adoptar otras representaciones,

especificando cuáles, sin que ello afecte a los procedimientos genéricos de resolución. Por último, el diseñador debe indicar las acciones que deben realizarse para resolver cualquier ejercicio basado en este ejemplo, es decir, con su misma estructura.

Para ejecutar los tres pasos que acabamos de detallar el sistema proporciona los medios necesarios para que el profesor pueda definir y clasificar los ejercicios que desea que constituyan el curso que haya planificado.

La tabla 1.1 muestra un ejemplo del tipo de datos que manipula *MathEdu*. En ella estamos especificando a alto nivel los elementos que constituyen el ejemplo anterior. Corresponden a los datos que se definen en el primer paso comentado más arriba.

definición
tipo de ejercicio → **integración**
enunciado → Cuaderno("Calcula la siguiente integral", *integral*)
integral → **Integrar(Sen(x), x)**
estrategia → **inmediata**

Tabla 1.1. Ejemplo de definición de un ejercicio.

Los elementos descritos son:

- *tipo de ejercicio*: indica que el ejercicio modelado corresponde a un ejercicio de integración.
- *enunciado*. Contiene la representación simbólica de un elemento propio de *Mathematica* denominado **Notebook** (cuaderno), el cual representa la ventana que se muestra al usuario y que va a ser utilizada como interfaz. Inicialmente sólo contiene los datos del enunciado agrupados en dos componentes: texto y fórmulas del enunciado.
- *integral*. Es una variable cuyo contenido corresponde al de la fórmula presente en el enunciado. Se determina dinámicamente durante el diseño del ejercicio por parte del profesor. Posteriormente, durante la resolución de los ejercicios generados a partir de este, *integral* tomará un valor concreto cada vez, diferente en cada ejemplo mostrado a cada alumno que utilice la aplicación de resolución interactiva de ejercicios.
- *estrategia*. Este campo hace referencia al modo de resolución de este tipo de ejercicio catalogándolo dentro de una jerarquía concreta.

A continuación, el segundo paso especificado le permite al diseñador generalizar los datos del ejercicio (tabla 1.2)

Generalización			
metavariables	valor	condición	generador
integrando	Sen(x)	TrigQ	simpleTrigGenerator

Tabla 1.2. Ejemplo de generalización de una variable.

El único elemento descrito es:

- *integrando.* Representa una metavariable. Este tipo de variables se utilizan para generalizar los contenidos de los enunciados. Su mayor particularidad es que llevan asociada información relativa a la expresión que generalizan. Dicha información se especifica mediante los campos:
  - *valor.* Valor dado al ejemplo inicial.
  - *condición.* Especifica la condición genérica que ha de cumplir la expresión generalizada. En este caso TrigQ hace referencia a que la expresión generalizada debe ser de tipo trigonométrico.
  - *generador.* Especifica la función que debe generar expresiones compatibles con la *condición*, en este caso cualquier función trigonométrica.

    Un ejemplo de función generadora es la que al invocarla nos devuelve un polinomio con el grado pedido y, opcionalmente, con término independiente o sin él. Combinando estas funciones podemos generar, por ejemplo, expresiones del tipo:

$$\frac{2x-3}{x^2-2x+5} \quad \text{ó} \quad x^2e^x$$

    las cuales no son resolubles de forma inmediata, como especifica la estrategia descrita en la tabla 1.1 y, en tal caso, deberían formar parte de otras estrategias de resolución. El hecho que pretendemos mostrar ahora es que es posible generar expresiones compuestas a partir de funciones generadoras de expresiones más simples.

Es importante señalar ahora que una vez hecha la generalización el valor de *integral* en la tabla 1.1 se transforma en

Integrar[*integrando*, *x*]

Esta expresión nos da idea del significado de dicho proceso de generalización: se ha sustituido la expresión simbólica que corresponde al ejemplo a partir del cual se está especificando un modelo general para ejercicios con características similares.

Por último, en el tercer paso descrito el diseñador especifica las acciones que debe realizar el alumno para desarrollar la resolución del ejercicio

acciones de resolución		
tipo de acción	descripción de la acción	
introducir expresión	**mensaje**	"introduce la solución"
	valor esperado	integral
introducir expresión	**mensaje**	"deriva la solución para comprobar el resultado"
	valor esperado	integrando

Tabla 1.3. Ejemplo de acciones de resolución.

En el caso concreto del ejemplo que estamos mostrando, como corresponde a una integral inmediata, resolvemos el ejercicio con dos acciones. Ambas acciones son idénticas en su configuración

- *introducir expresión.* Corresponde a una acción que pide al alumno que teclee una expresión. Dicha expresión debe adecuarse con la que el profesor haya predefinido. Este tipo de acciones se especifican mediante los dos siguientes campos
  - *mensaje.* Contiene el mensaje que se muestra al alumno para que realice la acción.
  - *valor esperado.* Contiene la expresión cuyo valor ha de coincidir con el correspondiente que teclee el alumno.

En la primera acción se pide al alumno que introduzca el resultado de la integral (es inmediata). Dicho resultado debe coincidir con el valor contenido en la variable *integral.* En la segunda acción se pretende comprobar que el resultado introducido es correcto. Para ello se le pide que lo derive, debiendo coincidir dicha derivada con la metavariable *integrando.*

Desde el punto de vista del profesor, *MathEdu* le exige, previamente a la realización del proceso de diseño, analizar con detalle aspectos propios de los ejercicios que desea modelar, tales como:

- qué estrategia se asocia al ejercicio,
- qué partes del ejercicio son susceptibles de generalización,
- qué variables deben representar dicha generalización,

- qué variable o variables contendrán la solución, o soluciones parciales, del ejercicio,
- si se puede reducir el ejercicio a otros problemas más sencillos,
- qué acciones considera relevantes para que sean mostradas al alumno durante la resolución, etc.

La forma en que el diseñador especifica estos datos debe ser suficientemente intuitiva como para no tener que exigirle conocimientos profundos de programación. Esto se resuelve en *MathEdu* con un entorno similar al que usará el alumno y que facilita en la medida de lo posible este proceso de definición de modelos de ejercicios a que nos referimos. Esta técnica de programación se conoce como *Programación por demostración.*

La primera conclusión evidente que podemos extraer del proceso de especificación anterior es que no es, en absoluto, trivial. A pesar de la sencillez del ejercicio propuesto, el número de pasos que el diseñador debe dar es considerablemente elevado. Más importante aún es el hecho del grado de generalidad que es necesario alcanzar para que la herramienta, en el momento de su utilización por parte del alumno, sea capaz de generar y reproducir de manera lógica los pasos necesarios para la resolución del ejercicio. Pensemos que el diseñador está "enseñando" al sistema a actuar de una manera específica ante cada ejercicio concreto a partir de las instrucciones genéricas que ha detallado para un ejemplo. Acabamos de referirnos en el párrafo anterior a esta técnica de programación que requiere un alto grado de abstracción por parte del diseñador y que, desde el punto de vista de la interfaz, supone un notable incremento de la dificultad en la creación de un entorno suficientemente intuitivo y de fácil manejo para el profesor.

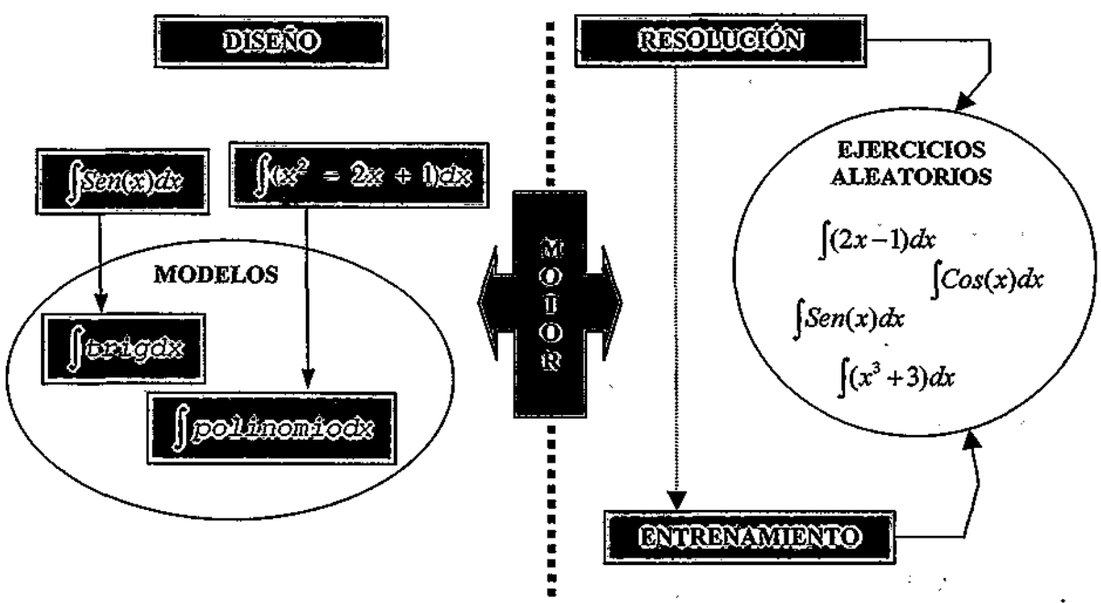

Figura 1.1. Módulos en *MathEdu.*

La figura 1.1 muestra esquemáticamente los tres módulos fundamentales existentes en *MathEdu* y sus interrelaciones. Se describe, de un modo muy elemental, cómo a partir de los datos de ejemplos el diseñador obtiene modelos abstractos asociados a ellos. A partir de estos modelos, los módulos de interacción con el alumno (resolución y entrenamiento) hacen uso de los mismos para dar lugar a ejemplos generados aleatoriamente a partir de aquellos. Existe, además, la posibilidad de colaboración entre el módulo de resolución y el de entrenamiento, como pretendemos reflejar mediante una conexión entre ambos.

El proceso de diseño debe producir como resultado un programa interpretable por el motor de resolución de *MathEdu*. El motor debe ser capaz de, a partir de la estructura de datos generada por el diseñador, interpretar dichos datos como si de un programa se tratase. Dicha interpretación de la información producirá, como consecuencias inmediatas

- una generación dinámica de valores correspondientes a las metavariables utilizando para ello la biblioteca de funciones generadoras y respetando las condiciones impuestas por el atributo *condición*,
- un diálogo con el alumno a partir de las acciones predefinidas

El diálogo que se establece con el alumno proporciona, a su vez, otra serie de datos que el motor interpretará para inferir distintas actuaciones basadas en los mismos. El objetivo que el diseñador debe procurar conseguir mediante la definición de acciones es que estos diálogos ilustren a los alumnos y les orienten en los pasos o etapas en los que comúnmente suelen tener dificultades. Nuevamente aquí se pone de manifiesto la importancia de un diseño "de alto nivel" en la definición de los ejemplos, con un alto grado de abstracción y generalidad.

La descripción que acabamos de presentar para un ejercicio sencillo de integración es válida para otros ejercicios que involucren cálculo simbólico, tal y como indicábamos más arriba. Así, por ejemplo, *MathEdu* es capaz de tratar ejercicios de materias diferentes como pueden ser derivadas, integrales, límites, ecuaciones diferenciales, etc. Ejercicios típicos en cada una de las anteriores materias pueden ser los siguientes

Ejercicio 1. Calcula la derivada de la función

$$y = Log(4x+2)$$

Ejercicio 2. Resuelve la siguiente ecuación diferencial

$$y'' + \frac{y'}{x} = 4x$$

Ejercicio 3. Calcula la integral

$$\int Sen(x)\,dx$$

Ejercicio 4. Calcula la integral

$$\int x \cdot Sen(x)\,dx$$

Ejercicio 5. Calcula el límite

$$\lim_{x\to 0}\left[\frac{x \cdot Sen(x)}{1-Cos(x)}\right]$$

Ejercicio 6. Calcula el límite

$$\lim_{x\to\infty}\left[\frac{x^3-2x+2}{x^3+x}\right]^{x^2}$$

Ejercicio 7. Resuelve la ecuación diferencial

$$(x+y+1)dx+(x-y^2+3)dy=0$$

Ejercicio 8. Calcula la integral

$$\int \frac{dx}{\sqrt{x^2-1}}$$

Ejercicio 9. Calcula la derivada de la función

$$y=\frac{x^2-2x+1}{e^{-x}}$$

Ejercicio 10. Calcula el límite

$$\lim_{x\to 0}\frac{|x|}{Sen(x)}$$

En definitiva, podemos concluir la introducción de esta memoria resaltando que la herramienta que hemos desarrollado y vamos a presentar en los próximos capítulos supone un salto cualitativo notable en la concepción existente hasta el presente de los tutores informáticos y las herramientas de cálculo simbólico y, en general, los entornos informáticos de aprendizaje interactivo para educación Matemática.

El contenido del resto de la tesis es el siguiente. En el capítulo 2 hacemos una revisión general del **Contexto Tecnológico** en el que se desarrolla el trabajo. Incluimos también un amplio apartado relativo a *Mathematica*, herramienta de cálculo simbólico sobre la que está construido *MathEdu*.

El grueso de la memoria se recoge en el capítulo 3 en el que se describe la **Arquitectura de *MathEdu*** así como una explicación detallada de cada uno de los módulos que la componen. Por tanto hablaremos en este capítulo del **Módulo de diseño de problemas: *MathEdu Designer***, describiendo la parte correspondiente al diseñador, vista desde el punto de vista de una herramienta informática que le posibilita definir ejemplos tipo de diferentes ejercicios. Posteriormente presentaremos el **Módulo de resolución interactiva de ejercicios: *MathEdu Solver***, en el que describimos la aplicación orientada al alumno, al que permite seguir los ejercicios de los diferentes cursos que el diseñador haya creado previamente. Finalmente presentaremos el **Módulo de resolución guiada: *MathTrainer***. Mediante este módulo el alumno puede acceder a la resolución guiada de los ejercicios propuestos bien a petición propia o bien por iniciativa del sistema.

El capítulo 4 está dedicado a las **Pruebas** que se han realizado con el sistema, describiendo las mismas y los resultados obtenidos.

Por último, el capítulo 5 resume las **Conclusiones** que podemos extraer del desarrollo del presente trabajo y de la consiguiente investigación que ha dado lugar al mismo, así como las perspectivas de desarrollo futuro tanto del propio sistema como de otros proyectos de investigación vinculados a este.

Tras los capítulos correspondientes al desarrollo de la memoria se han incorporado una serie de apéndices que profundizan en aspectos técnicos que surgen durante la exposición del grueso del trabajo. Finalmente se incluyen referencias de publicaciones relacionadas con la presente memoria así como de otras publicaciones referidas en la misma.

Capítulo 2.

CONTEXTO TECNOLÓGICO

2.1. Introducción

En el capítulo anterior establecíamos los objetivos de la investigación que deseábamos acometer y cuyo desarrollo constituye la presente memoria. Parece conveniente efectuar, en primer lugar y antes de pasar a la exposición detallada del sistema *MathEdu*, una revisión de entornos de características similares al que hemos desarrollado.

En los últimos años se han dado pasos importantes en la superación de las dificultades en la manipulación de documentos matemáticos mediante la utilización de sistemas específicos para la programación en contextos científicos, como *Mathematica* y *Maple*. Las versiones más avanzadas de estos sistemas integran, junto con su alta capacidad de tratamiento simbólico de todo tipo de expresiones y gráficos, aspectos cada vez más sofisticados, típicos de interfaces de usuario convencionales, como diálogos, menús de elección, botones, animaciones, etc. que permiten que los documentos de trabajo sean documentos activos, controlados por programas subyacentes. Estos avances están dando lugar a una nueva generación de aplicaciones interactivas para la enseñanza de las Matemáticas y otras ciencias. Sin embargo, a medida que se añaden capacidades interactivas a estos sistemas para el desarrollo de aplicaciones genéricas de carácter científico-técnico, resulta más difícil su programación; esta dificultad se incrementa especialmente cuando se desarrollan aplicaciones para la enseñanza, dado el alto nivel de interactividad que precisan. Las técnicas más potentes desarrolladas en los últimos años para simplificar el diseño de interfaces de usuario genéricas (Szekely, 1996), (Puerta, 1998) se basan en la utilización de la programación mediante ejemplos y la programación por demostración (Cypher, 1993) , (Lieberman, 2000) para permitir que el diseñador de la interfaz trabaje en el mismo contexto que el usuario final, de manera que el diseño resulte mucho más intuitivo y el diseñador se pueda centrar con mayor facilidad en los aspectos más delicados del mismo. Más adelante, en este mismo capítulo, estudiaremos en qué consisten estas técnicas de programación.

Antes de pasar a describir algunos ejemplos que ponen de manifiesto estos avances, conviene decir también que se han producido importantes mejoras en la dirección de incrementar la inteligencia de aplicaciones relacionadas con las Matemáticas, basados en nuevas técnicas de lógica computacional; estos avances, sin embargo, se limitan a la

construcción de sistemas que compiten en la resolución de colecciones amplias de problemas, (http://sunjessen24.informatik.tu-muenchen.de/~tptp/), (Sutcliffe, 1997), sin que existan todavía aplicaciones de estas tecnologías a la enseñanza. En nuestra opinión, la utilización de mecanismos de deducción natural, semejantes a los utilizados por las personas (Castells, 1995) podría ser fundamental en el futuro para que los avances en este terreno tengan aplicaciones en sistemas de enseñanza asistida por ordenador.

Al comenzar nuestro trabajo tomamos como referencia de partida otros trabajos que trataban, de un modo u otro, aspectos que *MathEdu* debía tender a unificar. Esencialmente existen dos clases de programas para educación en Matemáticas: por un lado nos encontramos con cursos interactivos y sistemas de resolución de problemas como son *Wiley Web Test* (Orr, 1996), *Calculus@Mathematica* (Uhl et al, 1998) y otros relacionados como pueden ser *Internet Calculus Program* (Manfredi et al, 1998), y *Calculus Wiz* (Stroyan, 1998). Estos sistemas ayudan al alumno a transitar a través de diferentes aspectos del aprendizaje matemático tradicional. Son dos los modos en que actúan, bien planteándoles problemas y validando las respuestas dadas por los alumnos (como en el caso de los dos primeros) o bien proporcionando una ayuda interactiva relativa a aspectos más amplios del proceso instruccional (en los dos últimos). La segunda categoría de programas para educación tiene que ver más con los esfuerzos realizados en la línea de profundizar en el conocimiento del alumno a un nivel semántico así como en facilitar el desarrollo de materiales educativos mediante el uso de herramientas de autor.

Antes de pasar a otras consideraciones al respecto de *MathEdu*, vamos a describir brevemente los sistemas enumerados y la relación que *MathEdu* presenta con ellos.

Wiley Web Test ofrece una gran variedad de ejercicios relacionados con derivación e integración. Permite a los alumnos practicar a través de la web las veces que sea necesario con el fin de obtener un grado de destreza en la resolución de ejercicios suficiente. Una vez logrado esto el alumno puede acudir a las pruebas presenciales para obtener sus créditos. La funcionalidad del sistema está limitada a las respuestas que los alumnos dan a los ejercicios propuestos (por ejemplo el cálculo de una integral o una derivada), no existiendo ningún otro tipo de comunicación entre el alumno y el sistema. Desde el punto de vista de *MathEdu*, nosotros tratamos de que además exista una comunicación bidireccional alumno-sistema. *MathEdu* no se limita a plantear al alumno un ejercicio y a corregir su contestación, sino que va guiando al alumno durante todo el proceso de resolución del ejercicio.

Calculus@Mathematica utiliza el programa *Mathematica* para ayudar al alumno en las rutinas habituales del aprendizaje matemático. De igual modo, *Internet Calculus Program* pretende proporcionar a los alumnos a través de Internet los contenidos necesarios para estudiar cálculo de un modo autodidacta. Ambos sistemas constituyen buenos ejemplos de sistemas tutores en Matemáticas. Este es un campo de continua y rápida expansión. *MathEdu* tiene cabida en él dado que también ofrece al alumno la posibilidad de atender las

explicaciones sobre el proceso de resolución de distintas clases de ejercicios, con lo cual ejerce una función tutora. En este sentido cabe mencionar el producto denominado *webMathematica* que ha sido presentado recientemente por Wolfram Research Inc. Es un conjunto de herramientas que permiten insertar comandos de *Mathematica* en páginas HTML sin que exista, a diferencia de *MathEdu*, ningún tipo de diálogo entre el sistema y el usuario, ni evaluación de metavariables, comparación de patrones, etc. Está accesible en http://www.wolfram.com/products/webmathematica/ .

Calculus Wiz utiliza *Mathematica* como plataforma al igual que *MathEdu*. Es un conjunto de *cuadernos* que ayudan al alumno a resolver una gran cantidad de los problemas típicos de un primer año de cálculo. El alumno sólo necesita rellenar determinados campos y pulsar botones para resolver los ejercicios que plantee al sistema. Desde un punto de vista interactivo, probablemente es uno de los sistemas de enseñanza de Matemáticas actuales más avanzado. Sin embargo no es una herramienta. Su contenido es fijo y, salvo nuevas versiones de la aplicación, el usuario no puede incrementar el contenido del curso por sí mismo. Esto no sucede en *MathEdu*. *MathEdu* ofrece al profesor la posibilidad de crear sus propios contenidos mediante una herramienta de autor.

Finalmente, *PAT* como *PAT OnLine* (versión experimental para uso en la web), son sistemas orientados al estudio del álgebra básica. Permiten al alumno definir variables asociadas a un problema, las relaciones existentes entre las mismas y obtener sus valores mediante las correspondientes ecuaciones. Su utilidad se incrementa gracias a la herramienta de autor que incorpora la cual permite crear nuevos problemas. Esta familia de sistemas representa un avance notable y decidido en la dirección del tratamiento de información semántica de los problemas. Sus limitaciones tienen que ver esencialmente con la comunicación con el usuario, que no permite el uso de operadores simbólicos matemáticos, estando limitada a la definición de las variables de un problema y de las ecuaciones que las relacionan. Esta cuestión está resuelta en *MathEdu* gracias al uso de *Mathematica* como plataforma de desarrollo y a la inclusión de mecanismos para el tratamiento de información simbólica (*patrones*).

En definitiva, una comparación entre *MathEdu* y los sistemas presentados muestra que el dominio de aplicación de *MathEdu* está próximo a los programas del primer grupo (*Wiley Web Test, Calculus@Mathematica, Calculus Wiz*). Además incluye un prototipo de herramienta de autor, como *PAT*, y permite el desarrollo de cursos con un grado de interactividad y generalidad semántica muy elevado.

MathEdu es un sistema interactivo de enseñanza de Matemáticas que posibilita el aprendizaje de una persona en un contexto específico como es el del cálculo simbólico y que incorpora funciones propias de los sistemas tutores inteligentes. El uso de sistemas con características como las de *MathEdu* se hace atractivo en todo tipo de contextos educativos, pudiendo destacar como cualidades más relevantes:

- El adiestramiento trata de ser personalizado al alumno, con lo cual es posible obtener una mayor eficiencia en el proceso de aprendizaje. El alumno incrementa su motivación al tomar parte activa en el proceso de enseñanza. En estos últimos años los sistemas adaptativos están consiguiendo avances notables en el grado de personalización del adiestramiento. Citemos, a modo de ejemplo, el sistema TANGOW (Carro, 1999b).

- Permiten que el alumno realice su adiestramiento cuando él lo desee dado que el sistema está siempre disponible.

- Se puede lograr un aprendizaje de calidad, ya que los sistemas plantean al alumno situaciones reales de aprendizaje y pueden ir incorporando nuevos elementos instruccionales que vayan incrementando las capacidades de enseñanza del sistema.

Los epígrafes siguientes analizarán, en primer lugar, distintos aspectos relativos a la representación del conocimiento matemático, partiendo de un análisis del estilo matemático como estilo formal de comunicación de conceptos y conocimientos. En el epígrafe 2.3 describiremos los sistemas tutores inteligentes así como los aspectos de los mismos presentes en *MathEdu*. El epígrafe 2.4 planteará cuestiones acerca de la elaboración de sistemas de enseñanza asistida por ordenador, sus características y principales dificultades. Finalmente, los dos últimos epígrafes hacen referencia a *Mathematica*, programa de cálculo simbólico en el que nos hemos apoyado para la elaboración de *MathEdu*. Para concluir el capítulo analizaremos el paradigma de programación mediante ejemplos, utilizado en el diseño teórico de la herramienta.

2.2. Representación del conocimiento matemático: del estilo matemático a la representación del conocimiento.

En Matemáticas, como en otros muchos ámbitos científicos, la evolución a lo largo de la historia ha ido configurando unas formas concretas de representación del conocimiento subyacente a dicha disciplina. Los sucesivos autores han tratado de establecer unas normas que les permitan utilizar los conceptos de una forma precisa, sin ambigüedades. Este es precisamente uno de los problemas básicos de cualquier lenguaje, su posibilidad de error. El error se produce en diferentes situaciones, siendo la más habitual la correspondiente a que tanto el emisor como el receptor manipulan distintos subconjuntos de objetos del lenguaje (palabras). El caso más extremo es el de dos personas con diferentes lenguas tratando de comunicarse. En este caso el subconjunto de palabras comunes a ambos (la intersección de sus lenguas) es vacío y la comunicación es imposible.

Sin embargo, el matemático ha tratado de minimizar esta posibilidad y para ello, fundamentalmente durante el siglo pasado, ha construido un lenguaje formal, basado en la axiomatización y en la formalización de los conceptos.

Pero la existencia de un lenguaje formal no garantiza la facilidad de la manipulación conceptual. Separar ambos lenguajes, el artificial y el natural, parece pues algo imposible. En cualquier texto matemático es necesario combinar ambos estilos y es la experiencia del matemático la que debe lograr que el paso de un estilo a otro sea lo más preciso posible.

Podemos entonces considerar el Lenguaje Matemático como unión de los lenguajes ordinario y artificial considerando como propios del mismo elementos que en ocasiones poseen referentes propios en el lenguaje ordinario, como pueden ser las letras que denotan conjuntos numéricos N, Z, Q,...o números singulares como e, i, π, etc. También se usan grafías que en el lenguaje ordinario carecen de significado alguno, por ejemplo *lim* por "límite" o *dx* por "diferencial de x". Hay palabras propias del lenguaje ordinario que en el contexto matemático modifican su significado como *grupo, anillo, ideal, cuerpo,* etc. Y numerosos signos artificiales muchos de los cuales comparten significado con otras disciplinas científicas (fundamentalmente la Lógica y la Física) como son $\in \rightarrow \subset \forall \forall \exists$ etc. (Lorenzo, 1989).

Son numerosos los estilos de representación y comunicación de la Matemática utilizados a lo largo de la historia. Geométrico, poético, algebraico-cartesiano, axiomático, formal, semiformal, etc. son algunos de ellos. Todos estos estilos han tenido su momento y han sido útiles para los matemáticos de la época en que se utilizaron. En la actualidad el más usado (y el que más nos interesa en nuestro caso) es el semiformal, que combina el lenguaje usual (sólo en la medida estrictamente necesaria) con los signos artificiales.

Si reflexionamos por un momento en cómo la formalización ha sido uno de los procesos esenciales para la representación del conocimiento matemático podemos llegar a la conclusión de que sin el desarrollo y generalización de los estilos formales (basados en la Lógica y la axiomatización) en el siglo XIX, sería difícil pensar en una representación del conocimiento matemático a partir de cualquier otro estilo. Desde el punto de vista de la informática fue esencial en su momento que autores como Leibniz o Vieta fueran capaces de abstraer y pasar de los enunciados de problemas entendidos como poesía o como mística numérica a enunciados que representan el problema sin ninguna concesión externa a la retórica. Veamos, como ejemplo, este texto sobre la formación y construcción del icosaedro en la obra de Luca Pacioli *Divina Proporción*.

> Entonces, puesto que el cuadrado de *qn* es, por la tercera del decimotercero, quíntuplo del cuadrado de *qm*, el cuadrado de *pn* será también, por la decimoquinta del quinto, quíntuplo

> del cuadrado de *lm*, pues, por la cuarta del segundo, el cuadrado de *pn* es cuádruplo del cuadrado de *qn*, y también el cuadrado de *lm* es cuádruplo del cuadrado de *qm*, por la misma. Y el cuádruplo es al cuádruplo como el simple es al simple, según afirma la decimoquinta del quinto...

Los ordenadores son capaces de tratar problemas que van mucho más allá del cálculo aritmético corriente: cálculos matemáticos, clasificaciones, operaciones con conjuntos, tratamiento de textos de gráficos e imágenes, síntesis de la palabra, reconocimiento de formas, diseños o enseñanzas asistidas, traducciones automáticas, animaciones, accionamiento de máquinas herramientas, etc. Todos estos tipos tan distintos de acciones no serían posibles sin la capacidad de la programación de las máquinas, programas que simbólicamente representan, cada uno de ellos, un "algoritmo" para la resolución de una clase de problemas. En el diseño de los algoritmos es corriente usar estructuras que afirman que si *a* es verdadero, entonces *b* es verdadero. El condicionamiento de *a* por *b* depende de reglas o axiomas libremente formulados y de una simbología que permite efectuar un tratamiento lógico, un *cálculo simbólico*. Gracias a esta simbología se ha podido sustituir el lenguaje coloquial impreciso por otro de símbolos totalmente preciso, bien determinado. Desde Babbage y Boole hasta la era de los ordenadores se ha desarrollado la capacidad de representar las reglas de la aritmética, del álgebra de magnitudes, el álgebra de proposiciones y, finalmente, el álgebra de conjuntos y la lógica simbólica (Ifrah, 1998). Este proceso ha supuesto dos milenios, desde la Antigüedad en que filósofos griegos como Sócrates y Aristóteles utilizaron la regla del tipo: "Si ... entonces", elaborando así uno de los fundamentos de la Lógica Matemática y del razonamiento deductivo.

Son múltiples los paradigmas de representación del conocimiento empleados en sistemas tutores inteligentes. La mayoría provienen del ámbito de la Inteligencia Artificial. Podemos señalar aquí algunos ejemplos al respecto. Anteriormente hemos expuesto algunas notas sobre PAT, que emplea reglas de producción para describir cómo hay que hacer cada tarea. Por su parte, ANDES (Gertner, 2000) emplea Redes Bayesianas para tratar las múltiples fuentes de incertidumbre existentes en la ejecución de acciones por parte del alumno y el grado de conocimiento del dominio que posee. Inclusive, recientemente se ha adaptado el concepto de sistema multiagente a los ITSs (Masthoff, 1997). Sin embargo no es frecuente encontrar en la literatura trabajos específicos sobre tutores inteligentes orientados a la enseñanza de las Matemáticas. Los libros de actas de congresos como International Conference on Mathematics/Science Education and Technology (M/SET 99, San Antonio, EE.UU.) o 5th International Conference on Intelligent Tutoring Systems (ITS 2000, Montreal, Canadá) ponen de manifiesto la carencia de este tipo de sistemas en eventos en los que de algún modo deberían estar presentes. Hoy en día los mayores esfuerzos se focalizan en sistemas adaptativos y colaborativos cuyas características difieren en gran medida de las de un sistema como *MathEdu*.

2.3. Sistemas Tutores Inteligentes

Los Sistemas Tutores Inteligentes (*ITS, Intelligent Tutoring Systems*) han sido utilizados a lo largo de su historia en una gran variedad de contextos diferentes. Han tratado aspectos tan diversos como pueden ser la enseñanza, básica y superior, con tutores de todo tipo de materias como Matemáticas o Física, podemos referir ANDES, entorno dirigido de resolución de ejercicios de Física (Gertner, 2000); navegación aérea, para el adiestramiento de controladores aéreos como TRIO (*Trainer for Radar Intercept Officers*) (Ritter, 1988) o ATC (*Air Traffic Control Training System*) (Morrisroe, 1986); en medicina, para adiestramiento de personal sanitario NEOMYCIN (Clancey, 1981); en industria, para el adiestramiento de operadores de plantas industriales como INTZA (Gutiérrez, 1994) o para el adiestramiento en detección de averías como QUEST (*Qualitative Understanding of Electrical System Troubleshooting*) (White, 1986). Inclusive para el adiestramiento en programación en lenguaje C (Kay, 1994).

La evolución de los tutores informáticos o Sistemas de Entrenamiento por Ordenador (*Computer Based Trainning, CBT,* obsérvese que aún no se usa el calificativo *inteligente*), se produce desde mediados del siglo pasado, finales de los años cuarenta. Está ligada inequívocamente al desarrollo de los ordenadores. Cómo no, la Segunda Guerra Mundial aportó su granito de arena en esta historia debido a las necesidades científicas y militares generadas. Las causas principales pueden ser:

- La necesidad de analizar las comunicaciones secretas operadas por los alemanes y los japoneses. Problemas de cifrado. Se utilizan máquinas específicas denominadas *Colossus.*

- La preocupación por conseguir cálculos de mayor precisión que los proporcionados por las calculadoras analógicas para, por ejemplo, aumentar la eficacia de los cañones.

- La necesidad de resolver problemas de simulación

- La necesidad de resolver problemas complejos de localización, relativos al radar.

Inicialmente estos sistemas eran muy precarios debido a las limitaciones propias de los equipos existentes (*Bell Labs Relay Computer, Model I*; data de 1939 y se considera la primera calculadora analítica binaria del mundo (electromecánica). El ENIAC se terminó en 1945 (Ifrah, 1998)), y se limitaban esencialmente a modelos matemáticos que simulaban el comportamiento de distintos componentes y sistemas.

Pero a partir de los años cincuenta se comienza a pasar de simples simuladores numéricos a sistemas que ya incorporan un componente instruccional que dirige el proceso de entrenamiento (O'Shea, 1985). Estos sistemas siguen la metodología de la instrucción programada lineal (Kearsley, 1983). Esta metodología requiere que el material a enseñar se organice de modo que se maximice el número de respuestas del alumno, suponiendo que las respuestas incorrectas producen un refuerzo negativo del aprendizaje. Todos los alumnos reciben el mismo conocimiento y en el mismo orden, sin que las respuestas del alumno influyan en el orden de presentación *lineal* del material. Last (1979) describe un programa para enseñar alemán absolutamente lineal.

El paso siguiente a dar parecía evidente, ya que la instrucción lineal no potencia nada las habilidades de unos alumnos frente a otros. Así se desarrollan los llamados "programas ramificados". En estos sistemas, la elección del material instruccional y las preguntas que se realizan al alumno se determinan en función de las respuestas del alumno elegidas entre una serie de posibilidades. Esta nueva filosofía produjo sistemas mucho más efectivos; aunque el diseño de estas estrategias era difícil y resultaba muy complejo en entornos en los que el número posible de respuestas distintas del alumno es muy elevado. Como consecuencia se introdujeron los "sistemas de autor". Estos necesitaban una serie de estrategias generales de enseñanza para producir el árbol con todas las interacciones posibles. Como ejemplo de este tipo de sistemas citaremos CALCHEM (Ayscough, 1977), sistema para la enseñanza de química.

Con el objetivo de conseguir que fuera el propio ordenador el que generara el material necesario para la enseñanza a partir de combinaciones de otros problemas ya existentes se idearon los sistemas generativos. Estos sistemas ya tratan de adaptarse a las necesidades y a las dificultades particulares de los alumnos y mejorar en eficiencia desde el punto de vista de uso de la memoria. Inicialmente el objetivo fue ambicioso pero los sistemas que se produjeron limitaban la posibilidad generativa a estrategias de instrucción y práctica (*drill and practice*), pequeñas unidades de instrucción seguidas de ejercicios prácticos. Esta estrategia a menudo seleccionaba los ejercicios de forma aleatoria sin concordar con las características del usuario (Koffman, 1975).

Una cuestión importante a valorar es cómo se efectuaba la evaluación en los sistemas descritos hasta el momento. Fundamentalmente se limitaban a valorar numéricamente las respuestas dadas por los alumnos sin ningún juicio al respecto del entendimiento del alumno de la materia estudiada. Además otros problemas que presentan estos sistemas son

- El déficit de uso del lenguaje natural dificulta la interacción y la comunicación con el usuario.

- No son capaces de aprender ni de la materia que enseñan ni del usuario, tal y como lo hacen los tutores humanos. Esto es debido al tipo de conocimiento estático que manejan.

Estamos, por tanto, en el punto intermedio de la evolución hacia los sistemas inteligentes. Los sistemas descritos hasta el momento no incorporan elementos que puedan diferenciar el aprendizaje entre unos alumnos y otros. Por tal motivo, no dejan de ser meros sistemas de presentación de información y evaluación de las respuestas del alumno. Para lograr la diferenciación es necesario dotar a los sistemas de la capacidad de adaptarse a las características diferenciales de sus usuarios. Por eso es necesario tener en cuenta a los usuarios, esencialmente al alumno, y desarrollar sistemas que exploten el conocimiento de las capacidades de sus usuarios. La evolución en este sentido produce el tránsito entre sistemas tutores (*CBTs*) y sistemas tutores inteligentes (*ITSs*).

En aquellos años se argumentó que hasta que no se aplicasen técnicas propias de la Inteligencia Artificial no podrían resolverse este tipo de problemas (Carbonell, 1970). Las soluciones adoptadas en aquellos momentos (Self, 1974), (Sleeman 1982), proponen separar el conocimiento del dominio (cuál es la materia a enseñar), de la forma en que se enseña (estrategias para enseñar) y a quién se enseña (alumno), existiendo actualmente un amplio consenso en establecer una arquitectura estándar basada en estas tres componentes

- Modelo del Dominio o del Experto (qué se enseña)
- Modelo del Alumno (a quién se enseña)
- Tutor (de qué manera se enseña)

Estos tres modelos junto con la interfaz, dado el importante papel que desempeña como comunicador del sistema con el exterior, constituyen los cuatro módulos considerados más relevantes en un tutor (Wenger, 1987), (Biegel, 1989), (Whitaker, 1992). A continuación vamos a efectuar una descripción breve de cada uno de estos cuatro módulos.

Modelo del Dominio

Organiza el conocimiento a enseñar. Sirve como fuente de conocimiento que debe presentarse al alumno y provee un estándar para la evaluación de las actuaciones del alumno, comparando las respuestas del mismo con las soluciones correctas. Un elemento importante de este modelo es la elección del esquema de representación del conocimiento que mejor se ajuste a la aplicación. Los esquemas más habituales en *ITS* son sistemas de producción, sistemas basados en *frames*, representaciones procedurales, representaciones

basadas en la lógica y modelos cuantitativos y cualitativos utilizados principalmente para modelos de simulación.

En el caso de *MathEdu* el conocimiento a enseñar se organiza en una estructura de datos que aúna simultáneamente información estructural y semántica. El sistema dispone de la estructura semántica que describe los distintos ejercicios definidos por el profesor. La información correspondiente a dicha estructura se instancia en tiempo de ejecución y se propaga entre los distintos objetos que configuran dicha estructura.

Modelo del Alumno

Se usa para representar y evaluar los progresos del alumno en el aprendizaje del dominio, incluyendo información referente a los conceptos aprendidos, errores cometidos, probables conceptos erróneos, características generales del alumno, etc. Esta representación debe actualizarse tras cada acción del alumno, por lo que debe incluir alguna componente dinámica. Las formas más habituales de representar el modelo del alumno son el *modelo Overlay*, donde la representación se realiza como un subconjunto del modelo del dominio (Wenger, 1987); y el *modelo diferencial*, donde además de representar el conocimiento correcto adquirido se tiene constancia de las desviaciones producidas en el proceso de aprendizaje (Sleeman, 1982b).

En la versión actual que se dispone de *MathEdu* no existe por el momento un modelo del alumno. Sin duda es la pieza esencial, bajo nuestro punto de vista, de un tutor inteligente. *MathEdu* debe incorporar en un futuro no muy lejano este tipo de modelo. Es más, *MathEdu* es la pieza básica que nos va a permitir dialogar con el alumno, evaluar su desempeño de tareas a la hora de resolver un ejercicio y, a partir de ahí, tratar de establecer un modelo diferencial del conocimiento como se describe en (Díez 1997). Uno de los objetivos finales de la Tesis es el de proporcionarnos una herramienta para poder dialogar con el alumno y, por tanto, ser capaces de asistirle de una forma personalizada, llevando cuenta de cuáles son los ejercicios que ha realizado, en qué orden, en cuáles ha tenido problemas, en cuáles ha solicitado ayuda. Un ejemplo de este seguimiento que se pretende proporcionar puede ser el siguiente. Supongamos que un alumno que ya ha estudiado y practicado los temas de derivación e integración con un cierto éxito esta tratando de resolver la ecuación diferencial ordinaria siguiente

$$(1+e^x)y\cdot y' = e^x$$

Al ser una ecuación de variables separadas se puede expresar, dividiendo entre $1+e^x$ ambos miembros y sustituyendo $y' = \frac{dy}{dx}$, como

$$y \cdot dy = \left(\frac{e^x}{1+e^x} \right) dx$$

integrando ambos miembros se debe obtener la solución general de la ecuación, por tanto el alumno deberá resolver la integral

$$\int (\frac{e^x}{1+e^x}) dx$$

Supongamos que al tratar de resolverla no se da cuenta de que es una integral inmediata o que se puede resolver por sustitución. Es en este momento cuando el modelo del alumno debe hacer uso de los datos que tiene almacenados sobre el alumno y, como sabe qué tipos de ejercicios de integración ha practicado antes, puede sugerirle ejemplos de ejercicios similares, o recomendarle que repase un tema concreto de integración, etc.

Tutor

Este módulo es el encargado de dirigir la instrucción que va a llevar a cabo cada alumno. Por tanto debe ser el módulo que decida qué conceptos se van a enseñar, en qué orden van a presentarse, en qué momento permitir o interrumpir la actuación del alumno, cómo se va a presentar la información, etc. Podemos distinguir varios modelos de tutor. Por un lado están los tutores de *iniciativa mixta* en los que el control se comparte entre el alumno y el tutor a través de un intercambio de preguntas y respuestas. El modelo de *descubrimiento guiado*, en el que el alumno posee todo el control de la actividad, siendo él quien determina la presentación de la materia. Por último, en los sistemas de tipo *monitor*, es el sistema el que posee el control de la sesión de enseñanza en todo momento; además, adaptará sus acciones a las respuestas del alumno según considere conveniente.

MathEdu se aproxima a este último tipo de esquemas ya que es el sistema el que controla la sesión de aprendizaje guiando al alumno con las preguntas oportunas en la resolución de ejercicios. Los ejercicios son generados (en el caso de *MathEdu Solver*) por el propio sistema, que es el que realiza la selección aleatoria de los ejercicios que se muestran al alumno. En el caso de *MathTrainer* los ejercicios pueden seleccionarse tanto por el tutor como por el alumno. Conviene aclarar que *MathEdu Solver* también puede, en un momento dado, tomar el control de la resolución de un ejercicio y mostrarle al alumno cómo se realizan las acciones de resolución del mismo. Esto suele deberse, fundamentalmente, a errores reiterados del alumno en un cálculo concreto. No podemos hablar pues de la existencia de un modelo del alumno.

Interfaz

Este módulo supone el medio de comunicación entre el sistema tutor y el exterior (alumno o profesor). Es el encargado de traducir las salidas que proporciona el sistema a una representación inteligible por el alumno y de convertir las entradas del alumno a la representación interna que maneja el sistema. Podemos resumir en dos aspectos la importancia de este módulo en la composición de los tutores: el primero es que debe poseer unas cualidades de sencillez de uso y de resultar atractivo para que capte la atención y sea utilizado con confianza por el usuario. El segundo es que los progresos actuales en la tecnología multimedia están produciendo herramientas y aplicaciones cada vez más sofisticadas, con un alto poder de comunicación que influye de forma decisiva en el diseño de los *ITS*.

No es habitual que los tutores incluyan simultáneamente los cuatro módulos que acabamos de describir, aunque todos los referidos al comienzo de este apartado poseen al menos alguno de ellos. Esto se debe a la enorme complejidad de aunar en un mismo sistema esquemas de representación del conocimiento muy distintos y que guarden relación entre si. Generalmente los sistemas existentes sólo desarrollan alguno de los módulos comentados, aunque naturalmente es posible desarrollar también el resto de módulos. Como ejemplo podemos comentar INTZA, tutor inteligente para entrenamiento en entornos industriales. En dicho trabajo se describe un tutor inteligente que incorpora un módulo didáctico que establece el Plan Instruccional a ejecutar (corresponde al módulo del dominio descrito más arriba); un gestor del alumno que evalúa y actualiza el modelo del alumno y un gestor del diálogo que establece las relaciones entre el alumno y el instructor con el sistema (corresponde pues a la Interfaz) (Gutiérrez 1994).

Finalmente, para concluir este apartado queremos destacar los desarrollos de teorías cognitivas para el desarrollo de sistemas inteligentes. De un lado podemos señalar el trabajo de J.R.Anderson que desarrolló la teoría cognitiva denominada ACT (Anderson, 1983), llevada posteriormente a la práctica en tutores inteligentes para la enseñanza de Geometría y Álgebra, así como para programación en LISP (Anderson, 1990). Paralelamente a dicha teoría se desarrolló Soar. Soar es una arquitectura para la resolución de problemas basados en el conocimiento, para el aprendizaje así como para la interacción con contextos externos (Rosenbloom, 1993), (Newell, 1990). La arquitectura Soar permite trabajar en todo tipo de tareas, desde las más rutinarias hasta problemas extremadamente complejos sin un estado final predecible. Permite también representar y utilizar en cada contexto las formas más apropiadas de conocimiento procedural, declarativo, episódico e icónico. Soar aprende sobre cualquier aspecto de las tareas que representa y sobre la aplicación de las mismas. En otras palabras, Soar pretende servir de soporte a todas las capacidades requeridas por un agente inteligente (Laird 1993).

2.4. Acerca del diseño y desarrollo de entornos informáticos de enseñanza interactiva.

Acabamos de comentar en el apartado anterior cómo en los últimos años se ha producido un evidente enriquecimiento en las posibilidades de manipulación e interconexión de diferentes representaciones del conocimiento matemático, haciéndolas más flexibles y potentes. Dos son los motivos esenciales para este avance. De una parte, la representación del conocimiento. La aparición de lenguajes estructurados, las técnicas de orientación a objetos, la programación simbólica y procedural, han posibilitado la investigación y el desarrollo de herramientas que se adentran en la manipulación de objetos abstractos. Por otro lado las interfaces más flexibles, adaptables a los requerimientos del usuario y más próximas a la realidad cotidiana que representan (con gráficos, tablas, ilustraciones, vídeo, etc.), van logrando aproximar la complejidad del lenguaje formal matemático y sus elementos a los usuarios de este tipo de herramientas. Sin embargo aún queda mucho camino por recorrer. Fundamentalmente el problema surge cuando se tratan de interpretar fórmulas o cuando se intenta manipularlas sin considerarlas como meros objetos gráficos. En este sentido LEIBNITZ (Greg, 1998) es un buen ejemplo de cómo ya se han dado pasos importantes en la dirección de la manipulación del contenido de ecuaciones y fórmulas. En el caso de *MathEdu*, la utilización de *Mathematica* ha supuesto una inestimable ayuda para poder manipular la parte simbólica de los ejercicios. A pesar de las facilidades que proporciona el editor de ecuaciones de *Mathematica*, no ha sido tarea sencilla manipular expresiones simbólicas en un lenguaje interpretado que además utiliza simultáneamente formas distintas de representación de la información, como ya explicaremos más adelante. A pesar de esto *MathEdu* representa un paso decidido en la dirección de mejorar la interacción entre el usuario y el sistema como ya hemos dicho anteriormente. Hasta el momento, el grado de interacción en aplicaciones concebidas para la enseñanza de las Matemáticas era muy escaso. En los mejores casos el alumno podía efectuar distintas representaciones geométricas modificando algunos parámetros de los que dependen dichas representaciones como en *Calculus@Mathematica*. En experiencias como las mostradas en (Serna, 1999), el alumno puede igualmente observar, siempre que lo desee, el comportamiento de procesos dinámicos (como el del péndulo múltiple) mediante animaciones de los mismos, al objeto de comprender mejor determinados procesos físicos y su representación matemática. O también puede introducir expresiones matemáticas para responder a cuestiones concretas como en *Wiley Web Test*. Pero en el sentido de considerar un intento de diálogo entre el alumno y el sistema a escala simbólica apenas existen avances.

Para los profesores y los diseñadores de material docente es imprescindible enfrentarse con cuestiones que inevitablemente surgen desde esta nueva concepción de las aplicaciones. Fundamentalmente podemos considerar las siguientes, algunas de las cuales surgen como conclusiones del grupo temático sobre *Entornos informáticos de aprendizaje*

interactivo desarrollado durante el 8° Congreso Internacional de Educación Matemática (ICME-8), celebrado en Sevilla en 1996:

- ¿Cómo pueden los profesores evaluar el aprendizaje de los alumnos utilizando este tipo de aplicaciones?.
- ¿Cómo adquiere el alumno el conocimiento matemático cuando se utiliza en este tipo de entornos?.
- ¿Cuál es el coste para el docente derivado de la implantación y uso de estas nuevas tecnologías?. ¿Presenta más ventajas que inconvenientes?.
- ¿Cómo se introduce de forma efectiva en el aula?.

Estas cuestiones ponen de manifiesto el interés de todas las personas dedicadas al diseño de este tipo de sistemas para establecer un marco teórico que fundamente los sistemas informáticos de enseñanza interactiva a partir de elementos propios de la psicología, la didáctica, la computación y la Inteligencia Artificial. Las preguntas formuladas no son en absoluto triviales. Por poner un ejemplo analicemos brevemente la última. En el último Simposium Internacional de Informática Educativa, celebrado en Puertollano (Ciudad Real) en noviembre de 2000, se puso de manifiesto la dificultad de implantar este tipo de tecnologías en el aula. La causa fundamental suele ser la falta de medios o de personal. Los docentes suplen con esfuerzo y grandes dosis de buena voluntad estas carencias. Sin embargo se pueden encontrar algunos ejemplos de implantación en el aula. Citemos TANGOW (Carro, 1999b) o (Gallego, 2000).

Desde esta perspectiva, *MathEdu* presenta algunas limitaciones ya que en su concepción, como herramienta de autor para desarrollar cursos sobre Matemáticas, no se ha previsto ninguna técnica propia de la psicología o de la didáctica. La libertad de que dispone el diseñador es, tal vez, el aspecto que suple estas carencias. *MathEdu* permite que cada diseñador especifique el conocimiento que desea transmitir y la forma en que va a hacerlo a su plena conveniencia. De algún modo estamos verificando con *MathEdu* la máxima de que los sistemas informáticos de enseñanza interactiva deben proporcionar un espacio en el que explorar libremente un mundo virtual diseñado para que construyan un determinado conocimiento. *MathEdu* proporciona al profesor un alto grado de libertad en el diseño de ejercicios y al alumno le permite explorar distintas formas de solución siempre que sean compatibles con el ejercicio propuesto. No cabe duda de que la resolución de todo tipo de ejercicio conlleva una serie de acciones ejecutadas en una secuencia lineal. Desde este punto de vista *MathEdu* sigue los principios del condicionamiento operativo, cuya ley fundamental formula que durante el proceso de aprendizaje, cuando a la aparición de una operación le sigue la presentación de un estímulo de refuerzo, la intensidad del aprendizaje se incrementa (Skinner, 1938). Como en tantos otros modelos instruccionales

en *MathEdu* se enfoca la instrucción hacia el componente común presente en todo sistema de enseñanza: el alumno. El aspecto relevante en todo modelo instruccional es cómo el alumno adquiere el conocimiento. Tradicionalmente la aplicación de las teorías del aprendizaje al diseño instruccional proviene de la psicología del comportamiento y cognitiva (Hiemstra, 1994). También han sido de gran influencia los trabajos de Skinner, Ausubel y Bruner. Pero la influencia más reciente en la elaboración de modelos instruccionales proviene de la psicología constructivista, relacionada con el hecho de cómo los alumnos construyen el conocimiento a través de experiencias, estructuras mentales y creencias (Jonassen, 1991). Esta búsqueda de nuevas corrientes psicológicas, desde las teorías del comportamiento hasta el constructivismo, que ayuden al diseño instruccional ha provocado la proliferación de numerosos modelos nuevos. Uno de los primeros ejemplos al respecto de esta búsqueda fue LOGO, lenguaje de programación cuyas primitivas realizan gráficos permitiendo al usuario identificar el proceso causa-efecto de una forma muy sencilla. Otros ejemplos, mucho más modernos, relacionados con la geometría son Cabri-Géomètre (Bellemain, 1992), o The Geometer Sketchpad (Jackiw, 1992). Ambos son de similares características. Permiten efectuar representaciones geométricas euclídeas de una forma muy simple gracias al grupo de primitivas que incorporan, las cuales hacen referencia a objetos geométricos como punto, recta, perpendicular, mediatriz, altura, etc. El alumno utiliza estas primitivas para crear sus propias construcciones y explorar los resultados de la geometría clásica. Ya hemos presentado PAT, sistema que permite evaluar el proceso de aprendizaje de construcción de expresiones algebraicas como modelos de problemas enunciados en lenguaje natural. Por su parte *MathEdu* permite al alumno la libertad de escoger entre varias alternativas de resolución de un ejercicio y entablar el diálogo con el sistema referido a la estrategia escogida. En esta libertad de acción el alumno construye su propio conocimiento.

En estos últimos años han surgido nuevas tendencias en la concepción de los sistemas de enseñanza asistida. Estas tendencias consisten en unir a los entornos de manipulación potentes demostradores de teoremas, con el fin de que los alumnos puedan explorar propiedades, probar conjeturas y buscar ejemplos. Buena muestra de este tipo de sistemas es el software de álgebra computacional CoCoA (*Computations in Commutative Algebra,* programa creado en el Departamento de Matemáticas de la Universidad de Génova, Capani, 2000), que permite explorar teoremas elementales de geometría euclídea o CHYPRE (Bernat, 1996), entorno interactivo para la resolución de problemas de geometría, que ilustra de igual modo esta tendencia, proporcionando al alumno la capacidad de explorar libremente un problema, así como probar cualquier alternativa de resolución.

En relación con lo que acabamos de exponer aparece otra de las conclusiones del grupo de trabajo mencionado anteriormente y que hace referencia a que los entornos informáticos de aprendizaje deben proporcionar al estudiante medios para transmitir sus ideas sobre objetos matemáticos y sus relaciones e, incluso, su forma de razonar. Así, el principal

vehículo mediante el cual los docentes pueden reproducir sus métodos pedagógicos consiste en describir y utilizar escenarios de aprendizaje que integren entornos informáticos de enseñanza interactiva. Estos tipos de escenarios crean situaciones de aprendizaje que se basan en preguntas y exploraciones, enfatizando la consecución de conexiones matemáticas. *MathEdu* es coherente en esta línea dado que permite al alumno explorar la resolución de ejercicios de cálculo simbólico manipulando expresiones y proponiendo alternativas de resolución. Esta actividad le obliga a razonar y reflexionar sobre la materia antes de comunicarse con el ordenador, dando pie por consiguiente a nuevas situaciones de aprendizaje.

2.5. *Mathematica*

Para poder llevar a cabo la tarea de construir un sistema que diese cumplida cuenta del objetivo fijado para esta tesis, era necesario que dispusiéramos de un sistema de los denominados de *cálculo simbólico*. Existen hoy día en el mercado distintos sistemas de estas características: *Derive*, *Matlab*, *Maple*, *Mathematica*, etc. Nosotros optamos por el último debido fundamentalmente a dos motivos

i. posee la capacidad de realizar programación simbólica y, más importante si cabe, reconocimiento simbólico de patrones (*pattern matching*).
ii. es el sistema que se viene utilizando tradicionalmente en el grupo de investigación en el que trabajamos

A lo largo del presente epígrafe vamos a desarrollar las funcionalidades básicas de *Mathematica*, incidiendo en aquellos aspectos que son relevantes desde el punto de vista del desarrollo de *MathEdu*. Antes de desglosar con mayor detalle algunos de los aspectos característicos de *Mathematica* a los cuales haremos referencia más adelante a lo largo de la memoria, haremos un breve resumen del programa. Diremos aquí que son referencias obligadas sobre *Mathematica* (Wolfram, 1991), (Wolfram, 1999), (Glynn 1997) y *webMathematica* (accesible en http://www.wolfram.com/products/webmathematica/).

Un sistema de manipulación simbólica es aquel que utiliza expresiones simbólicas y patrones sobre los que se definen reglas de reescritura, construyendo un sistema de evaluación con un determinado criterio de parada. Es posible asignar a cada patrón un tipo (numero, átomo, cadena, etc.) y utilizar esta asociación en el proceso de evaluación. *Mathematica* es un sistema de manipulación simbólica y está, por tanto, especialmente adaptado para utilizar expresiones matemáticas. Se puede utilizar como calculador simbólico y numérico, como lenguaje de programación de alto nivel, como sistema de representación del conocimiento para campos científicos y como sistema de visualización para funciones y datos.

En *MathEdu* resultan esenciales las capacidades mencionadas de *Mathematica* para definir y estructurar los datos relevantes de los ejercicios aportados al sistema por el diseñador; para ser capaces de generar funciones que concuerden con las especificaciones de los ejercicios mencionados o para manipular las expresiones procedentes de entradas de datos del alumno en el sistema, entre las funcionalidades más relevantes. Estas características de *MathEdu* son consecuencia de la notable capacidad de *Mathematica* para combinar el lenguaje de programación simbólico con el uso de patrones. Por otro lado, *Mathematica* está capacitado para realizar cálculos simbólicos. Por ejemplo, la ilustración 2.1 muestra el cálculo de la integral de la función $\frac{1+\sqrt{x}}{x^2}$.

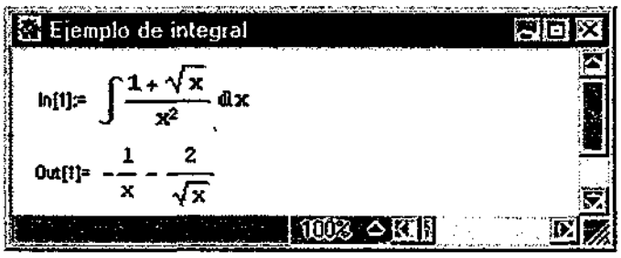

Ilustración 2.1. Ejemplo del cálculo de una integral.

En ocasiones *MathEdu* necesita realizar operaciones simbólicas como el cálculo de la integral anterior. En tal situación, *MathEdu* prepara la expresión simbólica que ha de evaluarse y el evaluador de expresiones de *Mathematica*, el *kernel* que vamos a describir a continuación, realiza esta operación de manera inmediata y automática.

2.5.1. El *front end* y el *kernel* en *Mathematica*

Mathematica está dividido en dos programas, el *front end* y el *kernel*, los cuales se comunican mediante un programa de comunicación de datos denominado *MathLink*.

El *front end* es la parte de *Mathematica* con la que interactuamos, manipulando ventanas, menús, cajas de diálogo, textos, gráficos, animaciones, sonidos, paletas, *notebooks* y celdas. El *kernel* es el programa que efectúa los cálculos que se le plantean desde el *front end*.

Las razones fundamentales de esta organización son, entre otras:

- Ambos programas pueden correr en ordenadores separados. Por ejemplo, el *front end* puede correr en un PC mientras que el *kernel* puede estar ejecutándose en un sistema UNIX en otro lugar distinto.

- El *front end* se puede conectar con distintos *kernels* al mismo tiempo, enviando y recibiendo cálculos de todos ellos simultáneamente.

- El *kernel* no dispone de interfaz. Es un programa puro en el sentido de que lee-datos/devuelve-datos. Por tal motivo es totalmente ejecutable en sistemas con diferentes sistemas operativos.

Insistíamos al inicio de este apartado en la capacidad de cálculo simbólico de que disponemos con un sistema como *Mathematica*. La ilustración 2.2 muestra algunas de las posibilidades del sistema.

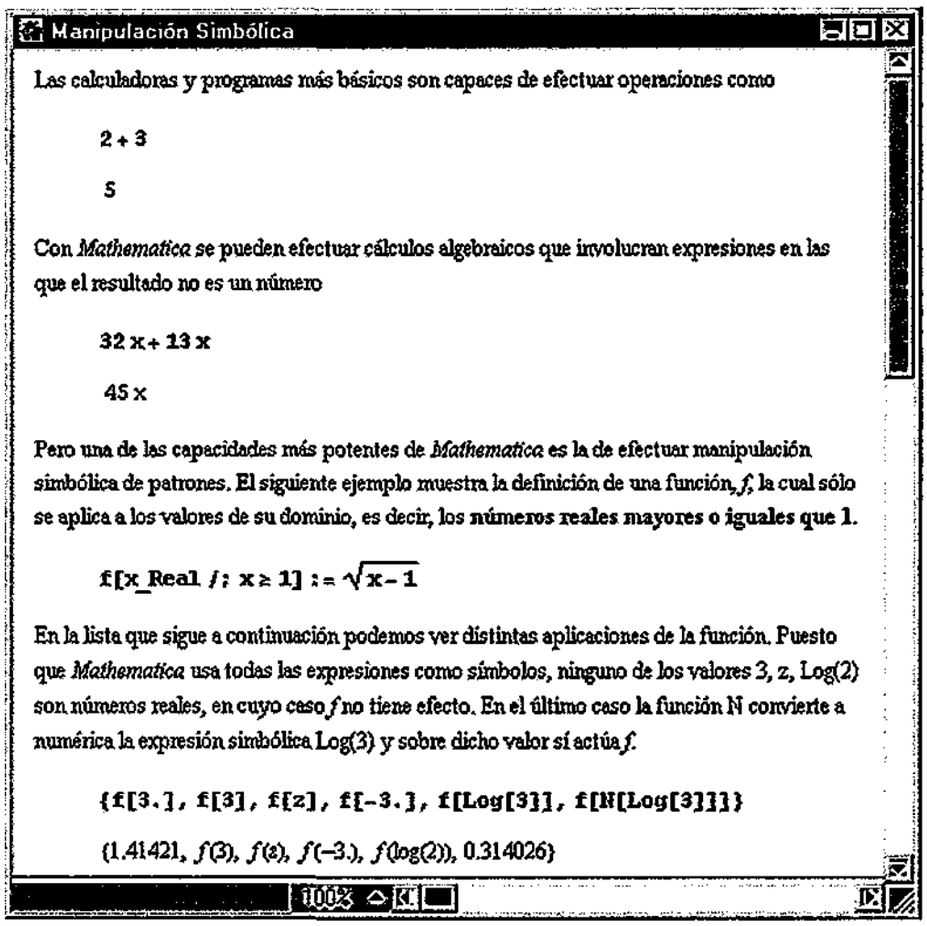

Ilustración 2.2. Ejemplos de manipulación simbólica con *Mathematica*.

En *MathEdu* hemos utilizado esta capacidad de cálculo para escribir procedimientos que interactúen con el alumno, efectuando, siempre que sean necesarias, las operaciones que se requieran para poder ir evaluando la resolución que hace de un ejercicio propuesto. Por este motivo, si consideramos importante la capacidad de cálculo, aun consideramos

más importante la posibilidad de programar simbólicamente en *Mathematica* y de efectuar *pattern-matching* con expresiones simbólicas como las anteriores.

No obstante, desde el punto de vista de la interfaz, a pesar del potencial de cálculo que acabamos de indicar, *Mathematica* no dispone de las primitivas adecuadas ni de un entorno de desarrollo suficientemente versátil que permita desarrollar interfaces compatibles con los estándares habituales utilizando elementos como menús desplegables, botones de selección exclusiva, campos con entrada de datos con opciones predefinidas, etc.

2.5.2. Programación en *Mathematica*; patrones simbólicos.

Podemos especificar dos tipos de programación en *Mathematica*: procedural y basada en patrones. En *MathEdu* vamos a combinar ambos tipos.

1. **Programación procedural**

En *Mathematica* es posible definir procedimientos dependientes de parámetros e invocar dichos procedimientos con objeto de obtener los valores resultantes de la ejecución del cuerpo de los mismos.

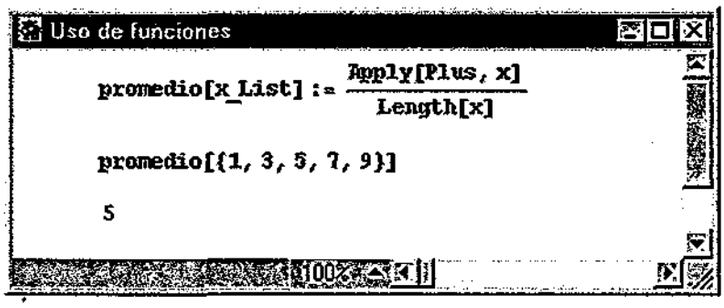

Ilustración 2.3 Ejemplo de uso de procedimientos en *Mathematica*.

Existe una gran cantidad de funciones definidas en el *kernel* de *Mathematica* que facilitan ambas formas de programación a que hemos hecho referencia. A modo de ejemplo la ilustración 2.3 muestra una función que sirve para calcular la media aritmética de cualquier cantidad de números. En este procedimiento es importante resaltar dos aspectos de la programación en *Mathematica*. En primer lugar el uso de patrones en los argumentos del procedimiento. Es decir, x_List, actúa como un patrón de lista y no admitirá ningún otro tipo de dato que no sea una lista. Los argumentos de los procedimientos en *Mathematica* son siempre patrones que hacen referencia a los tipos de datos que se desee. La segunda cuestión importante a resaltar es el uso del operador :=, el cual se interpreta como *asignación retardada*, que difiere de la asignación usual de valores a variables como puede ser

$$x = 2$$

En este segundo caso, desde el momento en que se evalúa dicha sentencia, x tiene como valor 2. A diferencia de esta asignación *inmediata*, la asignación retardada sólo se efectúa en el momento de invocar el procedimiento. En *Mathematica*, a diferencia de LISP, la evaluación se realiza mediante reglas de reescritura. Se aplican a cada expresión, mientras sea posible, todas las reglas disponibles.

Volviendo a la ilustración anterior, las funciones **Apply**, **Plus** y **Length** son funciones predefinidas en el *kernel* de *Mathematica* y, como puede comprobarse, la existencia de una biblioteca amplia de funciones facilita en gran medida la programación de cualquier tipo de procedimiento. En el ejemplo definimos el procedimiento (**promedio**) aplicando (**Apply**) la función suma (**Plus**) a los elementos de la lista (**x_List**) y dividiendo por el tamaño (**Length**, número de elementos) de dicha lista.

Antes de pasar a considerar los aspectos más relevantes de la programación basada en patrones que utiliza *Mathematica*, es necesario resaltar aquí una cuestión importante referente a la manipulación de expresiones simbólicas. En el *kernel* siempre se evalúa cualquier expresión que se envía, a menos que se le indique expresamente que no lo haga. Para prevenir una expresión de la evaluación existe un grupo de funciones (*Hold*, *HoldForm*, etc.) que han de anteponerse siempre a la expresión que se desea que permanezca sin evaluarse. A lo largo de la memoria se utilizarán en momentos puntuales estas funciones.

2. Programación basada en patrones

El mecanismo de reconocimiento de patrones de *Mathematica* es una de las herramientas más potentes del programa y le distingue de otros sistemas de Matemática simbólica. Para el objetivo del trabajo desarrollado en *MathEdu* ha sido una pieza esencial, sin la cual habría sido mucho más difícil abordar algunos de los problemas que han ido surgiendo en la manipulación de expresiones simbólicas.

En los párrafos siguientes vamos a tratar de exponer de forma somera una serie de ideas referentes a lo que es un patrón en *Mathematica*, los operadores habituales de los patrones, cómo se pueden construir funciones que utilicen los patrones, etc. con el fin de aclarar en la medida de lo posible conceptos que más adelante van a ser necesarios en el desarrollo de la exposición.

El uso de patrones más habitual es el de la declaración de funciones (como hemos visto en la ilustración 2.3).

Consideremos el siguiente ejemplo de declaración de la función *raíz* (ilustración 2.4). En el ejemplo se declaran dos definiciones de la función raíz. La primera acepta cualquier

expresión, devolviendo su raíz. La segunda acepta sólo valores estrictamente mayores que cero, devolviendo sus dos raíces (positiva y negativa). Al evaluar la raíz de un valor negativo actúa la definición de la primera declaración mientras que al evaluar la raíz del número 4 actúa la segunda declaración, pues se cumple la condición 4>0, obteniendo como resultado las dos raíces de 4.

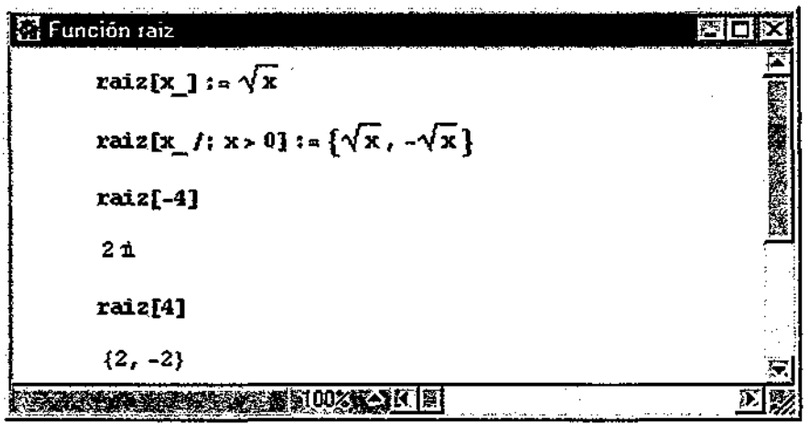

Ilustración 2.4. Ejemplo de declaración de funciones usando patrones.

En todas estas declaraciones de la función *raíz* se utiliza el patrón x_ que significa "cualquier expresión" y es la forma más sencilla de hacer referencia a un patrón de una expresión cualquiera. Obviamente las expresiones pueden ser mucho más complicadas y *Mathematica* nos permite referirnos a ellas mediante una segunda clase de patrones a la que denominamos *patrones estructurales*. Un ejemplo de este segundo tipo de patrones puede ser **y_List**. Una declaración de patrón como esa hace referencia a *cualquier expresión que sea una lista*, es decir se explicita la estructura que debe tener la expresión que reconozca el patrón. En consecuencia, la expresión

$$\{1, x\_, 2\}$$

sería válida para dicho patrón, mientras que

$$1+2x$$

no sería una expresión válida.

Otra forma de utilización de patrones es la que denominamos *patrones condicionales*. En este tipo de patrones se imponen condiciones (predicados que deben cumplirse) sobre las expresiones que se analicen. Un ejemplo de este tipo de patrón es **x_?IntegerQ**. Este patrón hace referencia a cualquier expresión que sea un número entero. Por tanto, el valor 2 sería reconocido por este patrón mientras que π no lo sería. La función de *Mathematica* **MatchQ** nos permite comparar expresiones con patrones estructurales o condicionales.

Así, en los ejemplos anteriores efectuaríamos las comprobaciones que se muestran en la ilustración 2.5.

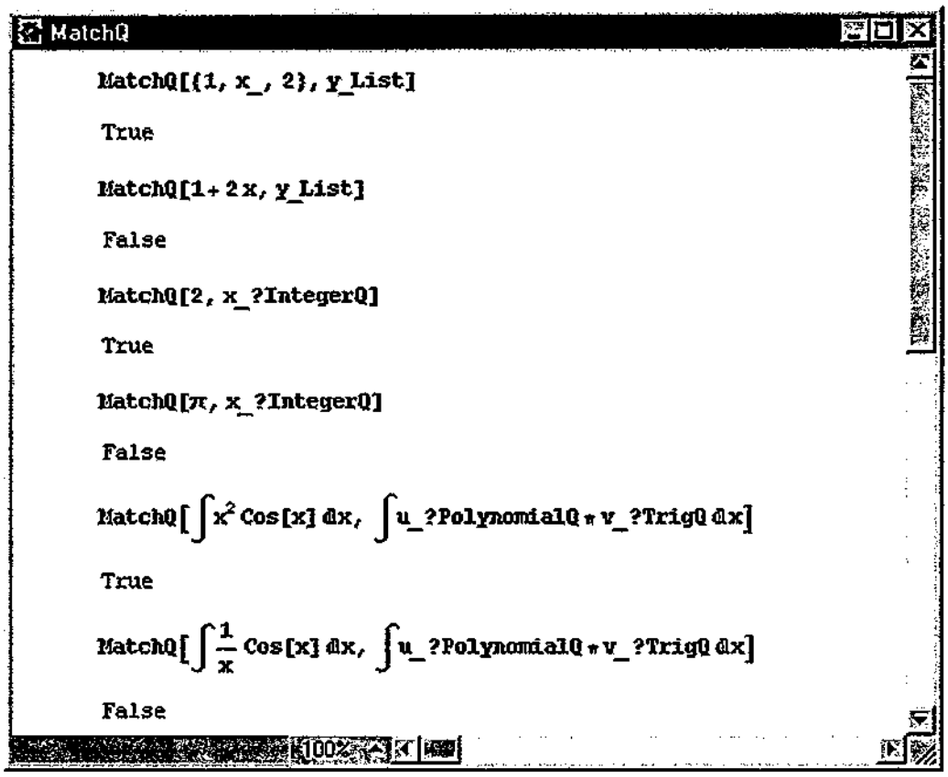

Ilustración 2.5. Uso de la función **MatchQ** con patrones estructurales y condicionales.

Hemos incluido dos verificaciones adicionales realizadas con la función **MatchQ** correspondientes a patrones condicionales que se utilizarán más adelante cuando expliquemos la generalización de ejercicios.

Con los tipos de patrones que acabamos de describir, combinándolos entre ellos para transformar expresiones en otras más complicadas, es posible efectuar programas simbólicos en los que nos hemos apoyado para el desarrollo de *MathEdu*. Para profundizar algo más en la programación simbólica basada en patrones vamos a mostrar una serie de ejemplos utilizando el mismo formato que *Mathematica*, indicando en una fila en negrita la expresión sin evaluar tal y como se escribe en el *front* end y en la siguiente línea la expresión evaluada procedente del *kernel*. Sea la siguiente expresión

a + b
a + b

Con el operador **ReplaceAll** (cuya forma abreviada es *expresión* **/.** *regla*), junto con la regla indicada con el operador **Rule** (cuya forma abreviada es *rhs* → *lhs*), podemos sustituir

la **b** de la expresión anterior. La orden debe leerse como "reemplazar con la expresión x^2 toda ocurrencia de **b** en la expresión **a + b**"

$$\mathbf{a + b\ /.\ b \rightarrow x^2}$$
$$a + x^2$$

La dificultad de la expresión no es obstáculo para poder utilizar la estructura correspondiente a la misma. Por ejemplo

$$\mathbf{a + Log(E^n)\ /.\ Log(E^x) \rightarrow x}$$
$$a + x$$

Mathematica localiza en una expresión compleja aquella subexpresión a que se está haciendo referencia en la regla, procediendo a su sustitución por el nuevo valor. Qué duda cabe que esta posibilidad, aunque útil, sería muy limitada si tuviésemos que ceñirnos siempre a expresiones que coincidiesen siempre literalmente. Es decir, si tratásemos de hacer algo como

$$\mathbf{a + Log(E^y)\ /.\ Log(E^x) \rightarrow x}$$
$$a + Log(E^y)$$

En este caso no hay *matching* de ninguna subexpresión y, por tanto, no se aplica la regla.

Mathematica nos permite generalizar el uso de las reglas mediante el operador **Pattern** (_). El patrón $\mathbf{Log(E^{y\_})}$ representa "cualquier expresión de esta forma, independientemente de lo que sea y_". Por tanto ahora sí que disponemos de toda la potencia de la aplicación de reglas sobre expresiones completas o subexpresiones. Un par de ejemplos

$$\mathbf{a + Log(E^{x+y})\ /.\ Log(E^{n\_}) \rightarrow n}$$
$$a + x + y$$

$$\mathbf{a + Log(E^{Sin(x+y)})\ /.\ Log(E^{n\_}) \rightarrow n}$$
$$a + Sin(x + y)$$

En el ejemplo que sigue observamos cómo *Mathematica* es capaz de *reordenar* las subexpresiones hasta encontrar una que coincide con la parte izquierda de la regla y, en consecuencia, la aplica. Es sencillamente genial

$$\mathbf{Cos(a)^2 + Cos(x)^2 + Sin(a)^2 + Sin(y)^2\ /.\ Sin(n\_)^2 + Cos(n\_)^2 \rightarrow 1}$$
$$1 + Cos(x)^2 + Sin(y)^2$$

Los patrones en *Mathematica* son estrictamente estructurales. Sólo hay dos casos en los que *Mathematica* reorganiza una expresión con el fin de buscar coincidencias con un patrón. En primer lugar, cambia el orden de los elementos en operaciones conmutativas como pueden ser la adición o la multiplicación. En segundo lugar, trata de efectuar agrupaciones en operaciones con la propiedad de asociatividad, como la adición y la multiplicación. Nunca tratará de efectuar reordenaciones como factorizaciones, aplicaciones de la propiedad distributiva, expansión de binomios, etc. a menos que el programador lo especifique.

Otra cuestión importante ligada a la programación basada en patrones, de la cual hemos hecho un uso profuso, consiste en la posibilidad de efectuar contrastes sobre las expresiones que manipulamos. Para realizar esto *Mathematica* dispone de la función boolena **PatternTest** (cuya forma abreviada es *pattern?test*). Aplicada sobre un patrón nos indica si dicho patrón cumple o no la condición impuesta. Consideremos las dos funciones siguientes

```
p1(x_?NumberQ):= x
p2(x_?IntegerQ):= x
```

la definición de **p1** se aplica cuando el argumento es un número de cualquier tipo. En el caso de **p2**, además de ser un número debe ser entero. La siguiente tabla 2.1 ilustra la aplicación de las dos funciones a distintos valores. Como se puede comprobar ninguna de las dos funciones aplicadas sobre **z** tienen efecto alguno. Esto se debe a que **z** representa un símbolo, no es una cantidad numérica.

valor	p1(valor)	p2(valor)
3.2	3.2	p2(3.2)
-4	-4	-4
z	p1(z)	p2(z)

Tabla 2.1. Ejemplo de aplicación de patrones sobre argumentos.

Esta posibilidad de trabajar de manera simple con condiciones estructurales sobre las expresiones ha resultado de gran utilidad e importancia en el desarrollo de *MathEdu*. Más adelante explicaremos con mayor profundidad aspectos más concretos relativos al sistema.

2.5.3. Representación simbólica de representaciones gráficas: las celdas en *Mathematica*.

Para concluir esta sección vamos a incluir una breve referencia al concepto de *celda* en *Mathematica*. Toda la información que se muestra en los *notebooks* de la aplicación se basa en el concepto de celda. Cada línea que escribimos en un *cuaderno de resolución* o en un *diálogo* para ser evaluada es una celda y, como tal, hay que construirla para poderla mostrar en el

lugar apropiado. La manipulación de expresiones es compleja, pero la posibilidad de utilizar símbolos matemáticos con total libertad es una de las ventajas que nos motivó a continuar con su utilización. En el Apéndice B explicamos someramente en qué consiste la manipulación de celdas. En él se trata de dar una idea de cómo se transforman las expresiones entre los dos programas principales de *Mathematica*, *front-end* y *kernel*, y cómo es necesario emplear en cada uno de ellos el formato adecuado al uso que se desea hacer de la expresión que se está manipulando, bien para ser mostrada al usuario o bien para su tratamiento simbólico.

2.6. Programación por demostración

Es probable que todos en algún momento hayamos vivido una situación como la que describimos a continuación. Imaginemos que todas las semanas utilizamos un programa comercial para anotar los cargos realizados sobre nuestra tarjeta de crédito. Los pasos a seguir, todas las semanas, son: efectuar la selección de la opción "Editar Transacciones" de un menú. Entonces un diálogo nos pregunta por la cuenta asociada y el rango de fechas para registrar los cargos. Buscamos entre una lista de 20 cuentas hasta encontrar "MasterCard". Escribimos en el campo correspondiente a la "fecha de inicio", por ejemplo la fecha de comienzo del mes, etc. Si este programa estuviera hecho específicamente para mí, probablemente tendría un botón etiquetado como "Añadir cargos MasterCard" que realizaría todos los pasos anteriores. Evidentemente este botón tiene sentido para mí, pero no para los otros cientos o miles de usuarios de un programa de estas características. En tal situación nos encontramos realizando un conjunto de tareas en vez de únicamente una y ello es debido a que utilizamos un programa de propósito general para realizar una tarea específica.

Esta situación es, en cierto modo, paradójica. Son los ordenadores los que deben efectuar las tareas repetitivas, las cuales "saben" hacer con gran eficiencia y, entonces, ¿por qué somos nosotros los que acabamos haciendo las tareas repetitivas en vez de los ordenadores?. Existen diversas técnicas denominadas como *programación de usuario* para permitir a estos definir sus propias preferencias. No son técnicas que requieran una programación en sí misma. Son técnicas que permiten alcanzar un objetivo propuesto sin necesidad de programar aunque para alcanzarlas debería efectuarse un programa. Pues bien, una de estas técnicas es la *programación por demostración*. Un ejemplo típico puede ser la declaración de macros en Microsoft Word. Otro ejemplo puede ser el siguiente: supongamos que tenemos un gráfico con cajas y textos dentro de las mismas, si cambiamos el tamaño de los textos y se salen de las cajas, hay que redimensionar todas las cajas para que los textos vuelvan a estar dentro. Si el número de cajas es elevado, se emplea un gran tiempo en una tarea que se puede automatizar y que resulta aburrida desde un punto de vista intelectual para la persona que la ejecuta.

Nos podemos referir en una primera aproximación a la programación por demostración como la programación en la interfaz del usuario. Para referirnos a una acción, únicamente hemos de realizar dicha acción. Programamos en el mismo entorno en el que vamos a ejecutar las acciones. Hemos de diferenciarla, por un lado, de la programación visual, en la que el predominio es de las acciones ejecutadas con el ratón, las cuales son interpretadas como parte del programa. Por otro lado, la programación por demostración también denominada programación mediante ejemplos ya que incluye aspectos propios de la programación usando ejemplos (por ejemplo macros) de las acciones repetidas que desea ejecutar. A diferencia de la programación convencional, más potente pero necesariamente mucho más técnica, estas otras técnicas de programación son mucho más intuitivas y requieren de una menor preparación técnica del usuario.

El tipo de tareas que un usuario de un programa suele desear hacer y en las que la programación mediante ejemplos puede resultar muy útil, podemos describirlas como:

a) Pequeñas tareas cotidianas que se pueden clasificar como preferencias, pero que el diseñador de la aplicación no tuvo en cuenta en su momento y en consecuencia no son accesibles desde los menús de preferencias predefinidos.

b) Automatizar tareas repetitivas en las que hay que efectuar una misma acción múltiples veces.

c) Poder construir miniaplicaciones que satisfagan requisitos específicos de los usuarios de los programas.

En definitiva, la programación mediante ejemplos debe verificar dos criterios básicos

1. El usuario especifica un programa para hacer una tarea concreta mediante los mismos comandos que utilizaría para efectuar dicha tarea manualmente

2. El usuario especifica un programa utilizando un ejemplo de lo que debe hacer la aplicación con los datos reales que esté manipulando actualmente. El sistema generaliza las acciones del usuario infiriendo la forma en que se debe ejecutar el programa incluso en un contexto diferente.

Existen numerosos ejemplos de programación mediante ejemplos. Se puede considerar como el primero de ellos **Pygmalion** (Smith, 1975a, 1975b), el cual es un entorno gráfico de programación. Fue desarrollado utilizando *Smalltalk*. Otro ejemplo interesante es **Tinker** (Lieberman, 1980). Se diseñó para ayudar a los programadores principiantes en los

problemas de demostrar condicionales y construir programas recursivos. Fue desarrollado íntegramente en Common Lisp.

Un clásico de la programación mediante ejemplos y muy útil desde el punto de vista de *MathEdu*, es **SmallStar** (Halbert, 1981). Halbert introduce la idea *programación en la interfaz del usuario*. Define el concepto de descripción de datos, el cual es básico para la comprensión de cómo los sistemas utilizan la programación por demostración refiriéndose a los objetos en las demostraciones efectuadas por los usuarios. Se utiliza un mecanismo de grabación de las acciones que ejecuta el usuario. En *MathEdu* también existen mecanismos de grabación (botones del menú *Designer* que describiremos en el capitulo 4) y se describen los datos mediante las metavariables. SmallStar se desarrolló utilizando *SmallTalk-80*.

Garnet y Amulet (Myers 1988, 1992, 1997) es un entorno de desarrollo de interfaces de usuario. Se basa en el paradigma objeto-instancia que utiliza representaciones de alto nivel de los objetos interactivos permitiendo establecer restricciones entre ellos. Garnet y Amulet incorporan herramientas interactivas basadas en programación por demostración para la construcción de interfaces de usuario. Con ellas se establecen restricciones gráficas en los objetos que sirven de instancia, y estas restricciones se generalizan, posteriormente, a los objetos en tiempo de ejecución.

The Geometer's Sketchpad (Jackiw 1992) es una aplicación para la enseñanza de geometría. Representa una forma muy novedosa de programación para el usuario final considerando como programación las construcciones geométricas realizadas en el entorno de desarrollo. Utiliza múltiples restricciones sobre los objetos geométricos del entorno de desarrollo.

TrIAs (Bauer, 2000) es un sistema para el entrenamiento de Agentes de Información, generando instrucciones que automatizan la extracción de información de páginas *web*.

En (Myers 2000) puede encontrarse una revisión de aproximadamente una docena de sistemas que abarcan diferentes dominios de aplicación y en los cuales el usuario define los comportamientos del sistema por demostración. Es interesante resaltar cómo se distingue entre sistemas en los se infieren propiedades de las generalizaciones hechas a partir de los ejemplos de aquellos sistemas en los que el usuario debe detallar explícitamente las propiedades que deben generalizarse a partir de los ejemplos. Como ejemplo del primer grupo de sistemas podemos citar Gamut (desarrollado en 1999), el cual emplea algoritmos propios de la IA como reconocimiento de planes o aprendizaje mediante árboles de decisión para crear juegos y aplicaciones interactivas. Como ejemplo del segundo grupo citaremos Topaz (desarrollado en 1998) para crear macros (guiones de acciones) en un editor de dibujo. Este sistema requiere que el usuario generalice explícitamente los parámetros de las operaciones que desee realizar.

Parte II

Diseño y resolución interactiva con MathEdu

Capítulo 3.

MathEdu

Una vez hecha una aproximación conceptual y terminológica al problema que abordamos, en el presente capítulo vamos a desarrollar con profundidad la estructura y las distintas funcionalidades de *MathEdu*. La idea es ir de los aspectos más generales de la aplicación, representados en la arquitectura de alto nivel, hasta los aspectos funcionales de nivel más básico, los cuales constituyen la base del sistema.

Antes de comenzar el desarrollo del capítulo queremos poner de manifiesto que en el campo en el que se desarrolla este trabajo, fundamentalmente la manipulación simbólica de objetos matemáticos, es notable la dificultad de lograr pequeños avances que representen algún tipo de novedad tecnológica. La dificultad que entraña obtener resultados significativos sobre sistemas de enseñanza asistida en Matemáticas no nos permite ofrecer al alumno sistemas con grandes novedades si no es a costa de invertir mucho tiempo en investigación y desarrollo.

El presente capítulo consta de cinco epígrafes bien diferenciados. El primero de ellos, que desarrollamos a continuación, tratará de ofrecer una visión global del sistema, describiendo las partes fundamentales que componen el mismo. Los tres epígrafes que se desarrollarán a continuación de la arquitectura, hacen referencia, cada uno de ellos, a una parte del sistema. El 3.2 nos introduce en la resolución de ejercicios con *MathEdu*. El epígrafe 3.3 desarrolla la herramienta de autor *MathEdu Designer*. El epígrafe 3.4 trata con profundidad acerca de *MathEdu Solver*. Estos dos módulos constituyen el sistema básico. Finalmente, el epígrafe 3.5 expondrá todo lo referente al valor añadido que tiene *MathEdu* con *MathTrainer*. Esta es una aplicación basada en el propio sistema básico y que posibilita la explicación de ejercicios o de errores cometidos durante la resolución.

3.1. Arquitectura de *MathEdu*

Como acabamos de exponer en la introducción del capítulo, a lo largo del presente epígrafe vamos a presentar cada una de las componentes de la estructura del sistema. En primer lugar describiremos la herramienta de autor o módulo de diseño de problemas.

Seguidamente presentaremos las dos aplicaciones de resolución interactiva de ejercicios. En los epígrafes siguientes haremos una descripción pormenorizada de cada una de estas partes, profundizando en la funcionalidad de cada una de las mismas.

La arquitectura de alto nivel de *MathEdu* es, conceptualmente, muy simple. Se compone de tres módulos que interactúan, no exhaustivamente, entre sí (es decir, no todos con todos), sobre la base común de *Mathematica*. Esta es la característica básica acerca de la estructura de alto nivel de *MathEdu*: todas las partes del sistema dependen de *Mathematica* para operar. El grado de interacción con el sistema varía de unos módulos a otros. En el caso del profesor/diseñador, la interacción va a ser muy estrecha e intensa, requiriendo con frecuencia que el profesor sea conocedor de mecanismos básicos de programación con *Mathematica*. Por el contrario, la interacción del alumno con el sistema va a ser mucho más sencilla. En el caso de la resolución interactiva, será necesario que el alumno vaya respondiendo las preguntas que le formule la aplicación. Si el alumno ha optado por la resolución guiada, simplemente se limitará a pulsar un botón de la interfaz con el que irá avanzando paso a paso en la resolución de un ejercicio.

Por tanto, a pesar de compartir una base común sobre la interfaz y el núcleo de *Mathematica*, las componentes de *MathEdu* requieren de grados de interacción hombre/máquina muy diferentes.

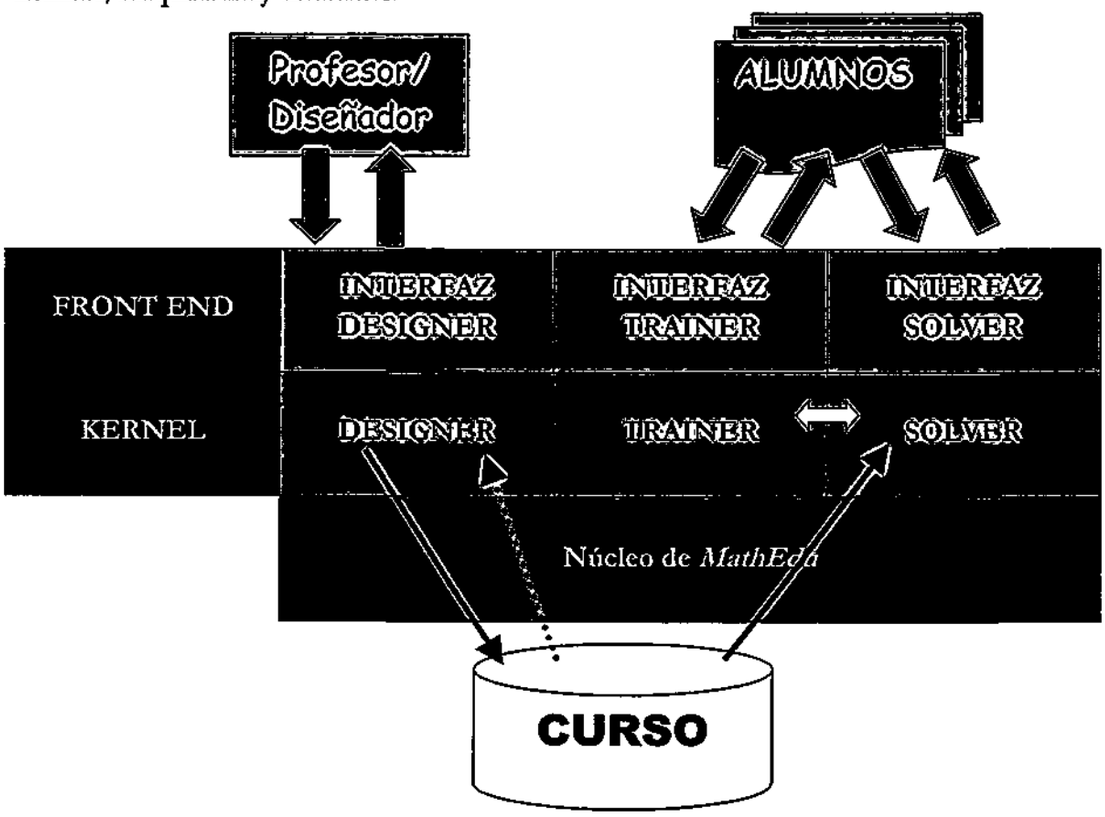

Figura 3.1. Arquitectura de *MathEdu*.

La figura 3.1 muestra de manera esquemática la estructura de alto nivel de *MathEdu*, diferenciando cada unos de los tres módulos básicos dentro del entorno común de *Mathematica*. Seguidamente vamos a describir esta estructura, detallando las relaciones existentes entre las distintas componentes.

Como consecuencia inmediata de que el sistema está desarrollado íntegramente sobre *Mathematica* existe la posibilidad de que en cualquier momento, tanto el diseñador como los alumnos tengan acceso al programa de cálculo simbólico a través de los cuadernos y paletas del *front end*.

3.1.1. La interfaz de *MathEdu*

Para comenzar a describir cuestiones específicas de los módulos que constituyen *MathEdu* es conveniente realizar algunas aclaraciones sobre las interfaces del sistema. Como acabamos de ver en la ilustración anterior, la interfaz entre los usuarios de *MathEdu* (tanto profesores en fase de diseño como alumnos en fase de resolución interactiva), y el propio sistema se utiliza el *front end* de *Mathematica*. Se pueden manipular objetos simbólicos, operar con ellos, transformarlos, realizar representaciones gráficas 2D y 3D, etc.

La comunicación entre *MathEdu* y *Mathematica* se realiza tanto en el nivel de la interfaz como en el nivel de evaluación de expresiones simbólicas (en el *Kernel*). Para la generación de los distintos elementos de la interfaz hemos elaborado una serie de funciones y procedimientos que residen, prácticamente en su totalidad, en el núcleo de *MathEdu* dado que son usadas indistintamente por los tres módulos del sistema. No obstante, las carencias específicas del *front end* hacen que la interfaz que hemos desarrollado disponga de escasos recursos. Gira, esencialmente, en torno a los tres elementos siguientes que se complementan a medida que se avanza en el diseño o resolución de los ejercicios:

a) Las **paletas de opciones**. Conjunto de botones agrupados en una paleta (ventana). Con ellas podemos elaborar menús de opciones con procedimientos asociados, los cuales se ejecutan al pulsar con el ratón sobre cada opción. El uso habitual de las paletas es el de proporcionar al usuario, profesor o alumno, una lista de opciones entre las que escoger un ítem. Los botones pueden hacer referencia a informaciones muy distintas, desde un *tipo de ejercicio* hasta un grupo de funciones, pasando por fórmulas, etc. En todos los casos el funcionamiento de la paleta es similar. Cuando se oprime uno de los botones con el ratón se ejecuta el procedimiento asociado al mismo. A modo de ejemplo, la ilustración 3.1 muestra la paleta del módulo *MathEdu Designer*.

Ilustración 3.1. Una paleta.

b) Los **diálogos** de introducción de datos. Son ventanas de diálogo con campos de introducción de datos y botones con distintas funcionalidades asociadas. Han resultado ser los elementos más útiles, auténticos canalizadores de información desde el usuario hacia el sistema. Permiten introducir cualquier dato, bien sea de tipo texto o con formato matemático pues, como expusimos en el capítulo anterior, *Mathematica* proporciona distintos formatos de representación simbólica así como paletas para introducir los símbolos gráficos habituales. *MathEdu* acepta todos los símbolos de las paletas a que acabamos de hacer referencia. Los datos que los usuarios teclean en los diálogos se leen y manipulan, según las necesidades de cada situación, gracias a la interacción de los botones que se incluyen con los mismos. La ilustración 3.2 muestra una ventana de diálogo de *MathEdu Designer.*

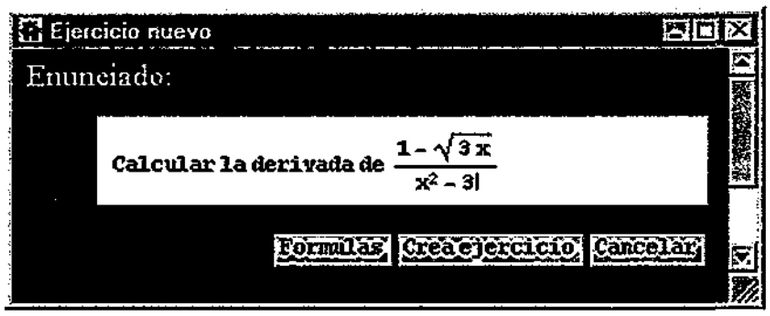

Ilustración 3.2. Ventana de diálogo.

c) los **cuadernos**. Son ventanas en las que se muestran datos y mensajes tanto al profesor en la fase de diseño como al alumno en la fase de resolución. Existen dos tipos de hojas de cuaderno:

- De *diseño*. Los *cuadernos de diseño* son los que utiliza el profesor durante la fase de diseño de ejercicios. En ellos se va recogiendo toda la información que el profesor va generando durante el diseño. En primer lugar se muestra el enunciado tal y como se ha definido. A continuación se van referenciando todas las metavariables que se declaren. Por último, se van detallando todas las acciones que el profesor defina para la resolución del ejercicio.

- De *resolución*. Los alumnos utilizan los *cuadernos de resolución* para la resolución de los ejercicios que se les propone. En ellos se les presenta el enunciado y se les van dando distintas indicaciones como pueden ser:

  - selección de una opción entre las de una paleta de estrategias o de acciones,
  - el dato o la fórmula que deben introducir,
  - mensajes de corrección/incorrección del dato introducido,
  - se les presentan resultados parciales que van obteniendo, etc.

Puesto que, además, los ejercicios propuestos pueden apoyarse a su vez en otros ejercicios más sencillos, se utiliza un cuaderno de resolución por cada ejercicio que se deba resolver.

En conclusión, podemos decir que la interfaz de *MathEdu*, aun no siendo especialmente vistosa, sí que resulta suficiente para establecer los diálogos necesarios entre el sistema y sus usuarios. Es obligado hacer una referencia especial a la facilidad con que se manipulan las fórmulas matemáticas gracias al *front end* de *Mathematica*.

3.1.2. Los módulos de *MathEdu*

MathEdu es un sistema completamente modular. Como ya hemos indicado anteriormente consta de tres módulos. Cada uno de los módulos dispone de una interfaz propia y un conjunto de funciones y procedimientos que constituyen su propio núcleo. Esta estructura modular se puede observar en la figura 3.1, en la que se muestran los tres módulos diferenciados que constituyen *MathEdu* integrados todos ellos en *Mathematica*. Todos los módulos comparten un núcleo común: el *Núcleo* de *MathEdu*.

La funcionalidad esencial de cada módulo es la siguiente:

a) *MathEdu Designer*. Es el módulo utilizado para la generación de los cursos. Este módulo es el utilizado por el profesor/diseñador. Las estructuras de datos creadas por el diseñador constituyen el CURSO, cuyo contenido puede corresponder, como vimos en el capítulo 1, a ejercicios de integración, derivación, ecuaciones diferenciales, etc. El contenido del curso siempre debe guardarse en un soporte físico como un fichero. Los distintos módulos acceden a él cuando es necesario. El contenido del curso sólo puede modificarse desde *MathEdu Designer*. Por tanto sólo el diseñador tiene la posibilidad de acceder a su contenido y modificarlo. Dicha estructura constituye la base de conocimiento sobre la que actúan los módulos de resolución que se describen a continuación.

b) *MathEdu Solver*. Este módulo es el que utilizan los alumnos para la resolución interactiva de ejercicios. Hace uso del curso creado por el profesor para generar ejercicios de enunciado aleatorio que han de ser resueltos de forma interactiva por los alumnos.

c) *MathTrainer*. Este módulo también es un módulo de resolución y, por tanto, utilizado por los alumnos. A diferencia del anterior, *MathTrainer*, no establece diálogo con el alumno, sino que le va mostrando, paso a paso, cómo es la resolución de un ejercicio. La selección del ejercicio puede realizarse bien al azar (por el propio sistema), o bien es

el propio alumno el que teclea el ejercicio del que desea obtener la correspondiente explicación.

Una característica importante de *MathTrainer* es que puede ser utilizado también por *MathEdu Solver* en aquellos casos en los que el alumno, durante la resolución de un ejercicio, solicita la ayuda del sistema ante una situación que no sabe resolver. En tal caso, *MathTrainer* nuevamente muestra al alumno la forma de proceder en la resolución del ejercicio propuesto.

Finalmente, el último supuesto en que interviene *MathTrainer* es en aquellos casos en los que durante la resolución interactiva de un ejercicio, el sistema detecta que el alumno comete reiterados errores en un paso concreto. En tal situación se interpreta que el alumno no es capaz de continuar y, de forma automática, el sistema interviene invocando a *MathTrainer* y tomando el control de la resolución desde ese momento.

Entre ambos módulos existe una comunicación bidireccional dado que *MathEdu Solver* accede a *MathTrainer* para poder dar ayuda y explicaciones a la resolución de ejercicios cuando el alumno lo requiere o comete reiterados errores en la resolución. Por su parte, *MathTrainer* accede a *MathEdu Solver* haciendo uso de parte de su funcionalidad para explicar la resolución guiada de ejercicios.

En resumen, concluimos este epígrafe sobre la arquitectura de *MathEdu* habiendo expuesto de un modo general la estructura básica del sistema y su integración dentro de *Mathematica*. En los epígrafes siguientes del presente capítulo vamos a ir describiendo con detalle la funcionalidad de cada uno de los módulos ya descritos, estudiando las interfaces de cada uno de ellos, las estructuras de datos que se generan así como su manipulación en la resolución de ejercicios y las distintas formas de interacción hombre máquina presentes.

3.2. Introducción a la resolución de ejercicios con *MathEdu*

Antes de introducirnos de forma extensa en el diseño de ejercicios y resolución (interactiva o guiada) de ejercicios con *MathEdu*, queremos mostrar de una forma concisa en qué consiste la parte correspondiente a lo que denominamos resolución interactiva. Recordemos aquí, como ya indicábamos en el capítulo 2, que los entornos de aprendizaje basados en el ordenador proporcionan al estudiante medios para transmitir sus ideas sobre objetos (matemáticos, en el caso de *MathEdu*) y sus relaciones e, incluso, su forma de razonar. El principal vehículo mediante el cual los docentes pueden reproducir sus métodos pedagógicos consiste en describir y utilizar escenarios integrados en sistemas informáticos de ayuda al aprendizaje, que se basan en preguntas y exploraciones, enfatizando la consecución de conexiones matemáticas (Allen, 1997). Mediante la integración de *MathEdu Designer* con *MathEdu Solver* tratamos de aumentar la integración de la tecnología en la educación.

A la hora de tratar la resolución de ejercicios mediante *MathEdu Solver* es imprescindible tener presente que el funcionamiento del mismo depende, en gran medida, del diseño de las acciones de resolución que debe realizar el profesor con *MathEdu Designer*. El proceso de resolución conlleva tres etapas que involucran simultáneamente al sistema y al alumno

1. Etapa de generación y comprensión del ejercicio

2. Etapa de planteamiento y selección de estrategias de resolución

3. Etapa de control y ejecución de las acciones de resolución

En cada una de las etapas anteriores intervienen tanto el sistema como el alumno. El sistema lleva el control de la resolución y el alumno razona y actúa a instancias de las cuestiones que el sistema le propone.

Cuando el alumno trabaja con *MathEdu Solver*, el motor de resolución encadena estas tres etapas con el fin de mostrar el enunciado del ejercicio generado y guiar el diálogo con el alumno y, por tanto, toda la resolución del ejercicio. La resolución de un ejercicio, planteada desde el punto de vista en que lo hacemos con *MathEdu Solver* es, en gran medida, similar a como se efectúa con lápiz y papel. Este proceso se ajusta a los pasos que acabamos de describir. Más adelante detallaremos cada una de las etapas enumeradas y los procesos que involucran.

Una vez que ha sido generado un ejercicio, el sistema concede plena libertad de elección al alumno para efectuar la segunda etapa. Le muestra las estrategias y le permite escoger la que él estime oportuno. Existe un módulo decisor (que se apoya en el uso de los patrones estructurales y condicionales del ejercicio generado) que se encarga de comprobar que la estrategia escogida es compatible con el ejercicio planteado y se puede utilizar para su resolución. Para cerciorarse de que el alumno ha escogido una estrategia "con conocimiento de causa", debe escoger también una descripción, del grupo de descripciones correspondientes a las estrategias presentadas, la cual ha de ser coherente con la estrategia que ha escogido.

Superados los dos pasos indicados que ubican al alumno frente a un modo de resolución del ejercicio que el sistema le ha planteado, este comienza a presentarle, de forma secuencial y tal y como el diseñador las especificó en su momento, las distintas acciones necesarias para alcanzar el objetivo de resolución del ejercicio propuesto. Se cumple así la tercera de las etapas referidas acerca de la resolución interactiva.

Con el fin de esquematizar los pasos descritos anteriormente y equiparar los pasos efectuados por *MathEdu Solver* con los efectuados por el alumno, vamos a mostrar en la tabla 3.1 dicha equiparación y los efectos que producen

MathEdu Solver	EFECTO	ALUMNO	EFECTO
Generación aleatoria	Determinación de *tipo*	Lectura del enunciado	Ubicación del ejercicio
Decisor de estrategias	Determinación de estrategia APTA para la resolución	Selección de estrategia de resolución	Comprensión del modo en que se resuelve el ejercicio
Gestión de acciones	Resolución del ejercicio	Ejecución de acciones de resolución	Ejercitación y comprensión del ejercicio

Tabla 3.1. Relación entre *MathEdu Solver* y el alumno.

Para clarificar en la medida de lo posible la tabla anterior, en las páginas siguientes vamos a mostrar cómo sería la resolución mediante lápiz y papel de un ejercicio de integración por partes. Una vez resuelto el ejercicio estudiaremos, desde el punto de vista del usuario de *MathEdu Solver*, el alumno, cómo la resolución electrónica sustituye a la resolución con lápiz y papel involucrando las tres etapas que de un modo sucinto acabamos de detallar. Finalmente vamos a imaginar una posible generalización del mismo por parte del diseñador, que nos introducirá en una terminología que será descrita con detalle en epígrafes posteriores.

3.2.1. Resolución de un ejemplo con lápiz y papel

Vamos a plantear cómo sería la resolución de un ejercicio de integración:

Ejercicio. Calcula la siguiente integral

$$I = \int 5x^2 \cdot Cos(x)dx$$

Resolución

Utilizaremos como estrategia de resolución de este ejercicio de integración el método de integración por partes. Se entiende que puede integrarse por partes porque la estructura subyacente (el patrón) del integrando corresponde al producto de una función polinómica por una función trigonométrica simple y dicho patrón nos sugiere la posibilidad de usar la estrategia indicada. Así consideramos

$$u = 5x^2 \qquad [1]$$
$$v = Cos(x)dx \qquad [2]$$

de donde obtenemos, derivando e integrando respectivamente las expresiones [1] y [2]

$$du = 10xdx$$
$$V = Sen(x)$$

la expresión general para la resolución por partes de una integral es

$$I = u \cdot V - \int Vdu \qquad [3]$$

por tanto, la integral inicial queda, realizando las sustituciones oportunas,

$$I = \int 5x^2 \cdot Cos(x)dx = 5x^2 \cdot Sen(x) - 10\int x \cdot Sen(x)dx \qquad [4]$$

De este modo hemos convertido la integral inicial en una expresión en la que hemos de resolver la siguiente integral más sencilla, la cual la resolvemos nuevamente por partes

$$I_1 = \int x \cdot Sen(x)dx$$

consideramos

$$u = x \qquad [5]$$
$$v = Sen(x)dx \qquad [6]$$

de donde obtenemos, derivando e integrando respectivamente las expresiones [5] y [6]

$$du = dx$$
$$V = -Cos(x)$$

por tanto, el resultado del ejercicio I_1 planteado es

$$\int x \cdot Sen(x)dx = -x \cdot Cos(x) + \int Cos(x)dx = -x \cdot Cos(x) + Sen(x) + C \ .$$

Sustituimos el valor de I_1 en [4] y obtenemos el resultado final de la integral pedida

$$\int 5x^2 \cdot Cos(x)dx = 5x^2 \cdot Sen(x) - 10\left(-x \cdot Cos(x) + Sen(x) + C\right) =$$
$$= Sen(x)\left(5x^2 - 10\right) + 10x \cdot Cos(x) + C$$

Con el fin de verificar que esta expresión es correcta vamos a proceder a derivarla, así

$$\frac{d(Sen(x)\left(5x^2 - 10\right) + 10x \cdot Cos(x) + C)}{dx} =$$
$$= Cos(x)\left(5x^2 - 10\right) + 10x \cdot Sen(x) + 10\left(Cos(x) - x \cdot Sen(x)\right) =$$
$$= 5x^2 \cdot Cos(x)$$

Una vez resuelto este ejemplo el profesor explicará la forma de generalizar un ejercicio de estas características para obtener un modelo que represente a cualquier ejercicio con la misma estructura y datos similares. Existen numerosos ejercicios con datos de las mismas características de los utilizados en este ejemplo y que dan lugar a ejercicios que se resuelven del mismo modo en que hemos efectuado la resolución de éste. Por ser más concretos, la resolución del ejemplo anterior es exactamente igual a la resolución de

Ejercicio. Calcula la siguiente integral

$$\int x^2 \cdot Sen(x)dx$$

O bien, usando el mismo procedimiento de resolución por partes simplificado, a la resolución de

Ejercicio. Calcula la siguiente integral

$$\int x \cdot Cos(x)dx$$

y de

Ejercicio. Calcula la siguiente integral

$$\int (x+1) \cdot Sen(x)dx$$

En consecuencia podemos concluir que para ejercicios de un mismo tipo podemos aplicar una estrategia común de resolución, independientemente de los datos concretos que aparezcan en el enunciado.

Hasta aquí nos hemos limitado a exponer lo que la teoría general de resolución de integrales indefinidas contempla para la resolución de este tipo de ejercicios. Desde el punto de vista del profesor, la resolución de ejercicios de estas características requiere de ciertas destrezas que el alumno ha debido obtener tras el estudio de la teoría. Básicamente podemos enumerar

- Ser capaces de identificar un tipo de ejercicio por sus datos,
- Ser capaces de decidir las partes del integrando que componen cada una de las funciones buscadas,
- Ser capaces de derivar correctamente funciones sencillas (polinomios en el ejemplo),
- Ser capaces de integrar correctamente funciones sencillas (senos y cosenos en el ejemplo),
- Conocer la teoría correspondiente a la resolución de integrales mediante la estrategia *por partes*.

A la vista de la riqueza de actividades y conocimientos que involucra la resolución de ejercicios con características como las de los mostrados en los ejemplos, parece claro el interés que puede despertar en el profesor un mecanismo que sea capaz de mostrar al alumno múltiples ejercicios como los anteriores. Sería deseable, además, que se pudiera establecer un cierto tipo de diálogo con los alumnos con el fin de explorar su grado de conocimiento de este u otros métodos de resolución de ejercicios. *MathEdu* es el mecanismo encargado de proporcionar al profesor la herramienta de autor para el diseño de modelos de ejercicios (generalización a partir de ejemplos) y al alumno la aplicación para la resolución interactiva o guiada de los ejercicios generados a partir de los modelos descritos. Veamos entonces, a partir del ejemplo anterior, cómo sería la resolución con *MathEdu Solver* de este mismo ejercicio y cómo sería la generalización en *MathEdu Designer* del ejemplo anterior.

3.2.2. La resolución con *MathEdu Solver*

Hemos comentado al inicio del epígrafe 3.2 que el proceso de resolución de ejercicios se realiza en tres etapas. La primera consiste en la generación de un ejemplo de ejercicio similar al usado para construir el modelo. Supongamos que el ejercicio a resolver es el mostrado en la ilustración 3.3, el mismo que hemos resuelto con lápiz y papel, al objeto de poder efectuar las comparaciones que sean oportunas.

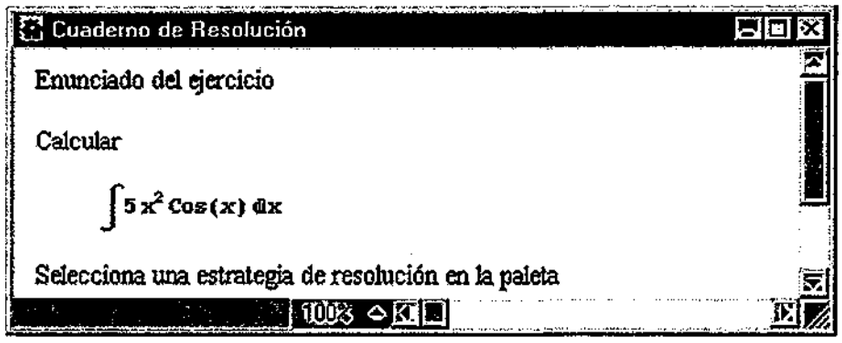

Ilustración 3.3. Enunciado del ejercicio a resolver.

Este ejercicio es de tipo integración y, en tal caso, hay diversas estrategias para resolver ejercicios de dicho tipo. Por ejemplo se pueden considerar, entre otras, las estrategias de resolución inmediata para funciones sencillas de tipo polinómico, logarítmico o exponencial, por cambio de variable, de funciones racionales, por partes, trigonométricas, etc.

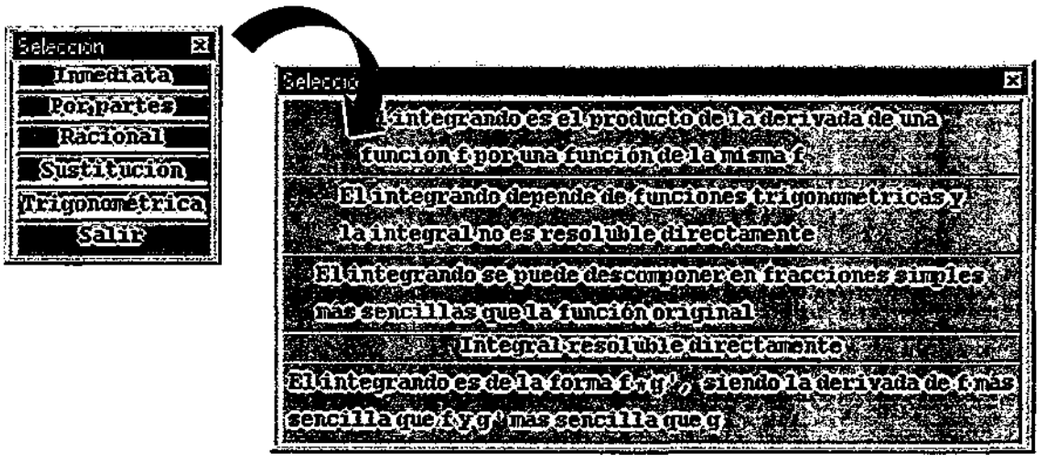

Ilustración 3.4. Paleta de estrategias y descripciones de estas.

El alumno debe escoger alguna de estas estrategias y justificar su elección seleccionando entre todas las descripciones de las estrategias indicadas aquella que considera relativa a la estrategia que ha escogido (ilustración 3.4). Cada estrategia de resolución lleva asociados patrones estructurales y condicionados para las fórmulas que intervienen en el enunciado.

MathEdu hace uso de dichos patrones para verificar que la estrategia escogida por el alumno es compatible con el ejercicio generado.

Las descripciones de las estrategias las establece el profesor durante el proceso de generalización de un ejercicio y se presentan al alumno, durante la resolución, como opciones excluyentes de un menú. En el caso de que la estrategia y la justificación que escoja sean apropiadas para resolver el ejercicio planteado (que no tienen por qué ser únicas, cuestión esta de gran trascendencia en la concepción de la aplicación), el sistema comenzará a mostrarle secuencialmente las acciones que el diseñador haya previsto para el tipo y estrategia involucrados en la resolución.

Las acciones que el diseñador ha especificado para el caso de una integral que se resuelve por partes involucran los aspectos a que nos hemos referido anteriormente acerca de selección de expresiones, derivación, integración y conocimiento de la teoría propia de esta estrategia. Para poder ir respondiendo las cuestiones que *Solver* plantea al alumno su interfaz está formada por ventanas de diálogo, paletas de selección de alternativas y un cuaderno de resolución en el que se van mostrando mensajes así como los datos que ha ido introduciendo en el sistema, junto con su validación. En las ventanas de diálogo el alumno teclea los datos que le pide el sistema, pudiendo introducir cualquier fórmula matemática. El aspecto de una ventana de diálogo a que nos referimos se muestra en la ilustración 3.5.

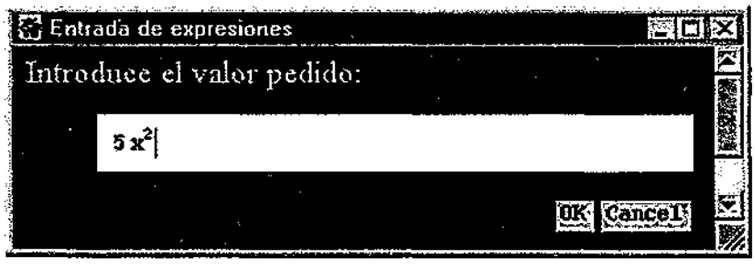

Ilustración 3.5. Diálogo de introducción de expresiones.

En ella se muestra un diálogo de introducción libre de expresiones mediante el cual el alumno introduce el valor de la función *u* que está declarando para poder resolver la integral por partes.

Mediante preguntas planteadas en el cuaderno de resolución y las oportunas respuestas del alumno a través de los diálogos, el sistema va evolucionando en la resolución del ejercicio produciendo como resultado un cuaderno que refleja todo el proceso seguido desde el inicio hasta la conclusión del mismo. Como se puede apreciar en la ilustración 3.6, en el cuaderno de resolución se le van mostrando mensajes de lo que se le pide y debe ir haciendo. En concreto se aprecia el estado del cuaderno tras haber respondido a las cinco primeras preguntas relativas a la definición de las partes que el alumno desea establecer (la primera repuesta corresponde a la ilustración 3.5). Las respuestas que el alumno ha dado

han sido correctas. *MathEdu Solver* las ha verificado en el momento de ser tecleadas por el alumno. Las respuestas del alumno corresponden a distintos tipos de acción utilizados por el diseñador durante la elaboración del ejercicio. Las acciones que el profesor ha usado en este caso concreto llevan asociadas, en el momento de la resolución interactiva por parte del alumno, la comprobación de que la expresión introducida es igual (o equivalente) a la que el profesor ha definido en el diseño.

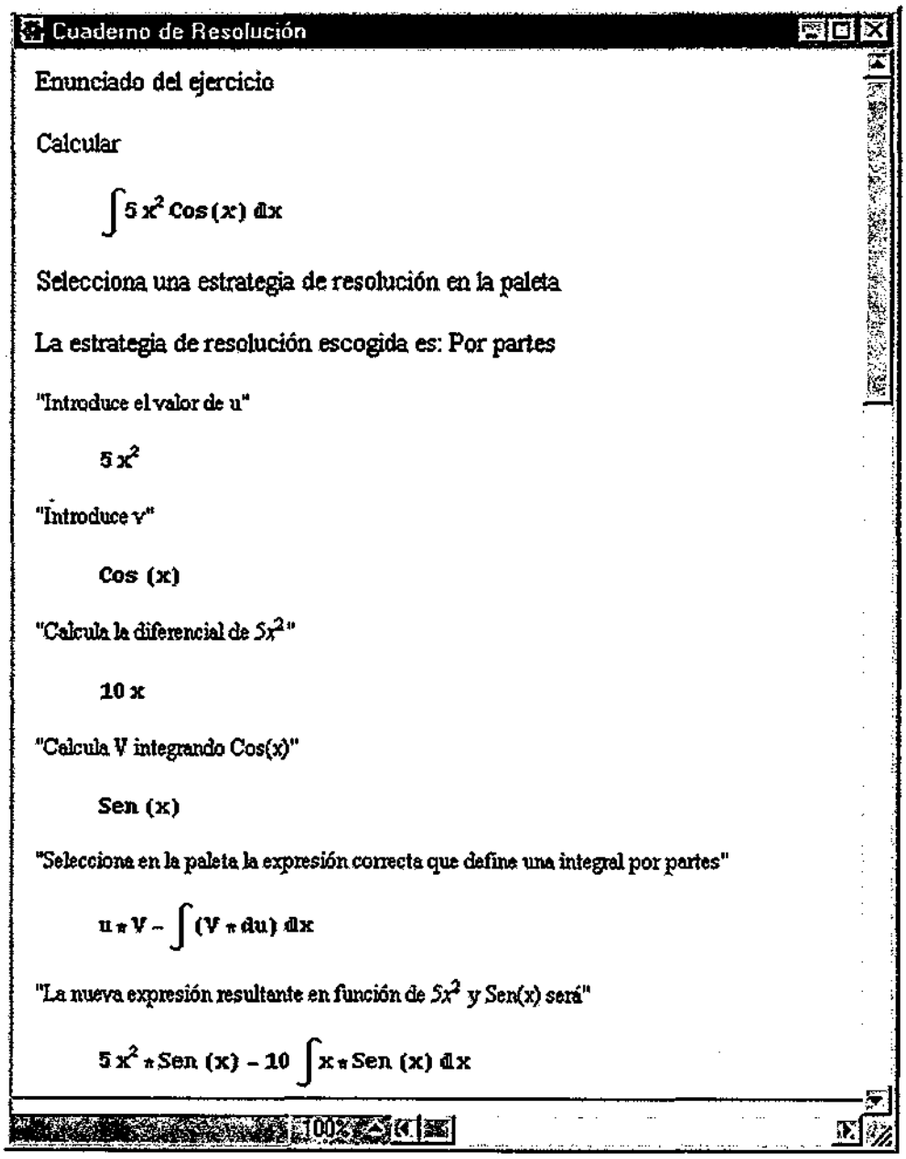

Ilustración 3.6. Cálculos efectuados en la resolución del ejercicio.

Otros tipos de acciones, como veremos en el epígrafe siguiente, efectúan otras formas de verificación o, inclusive, no hacen comprobación alguna.

En el caso concreto de este ejercicio, la resolución de la integral que se plantea requiere resolver sendas integrales más sencillas que la original, como subproblemas del mismo. Estos son:

Ejercicio. Calcula la siguiente integral

$$\int Cos(x)dx$$

y

Ejercicio. Calcula la siguiente integral

$$\int x \cdot Sen(x)dx$$

La primera de estas integrales el profesor se la plantea al alumno como una expresión *inmediata* que debe introducir después de haber realizado el oportuno cálculo. Sin embargo la segunda se plantea como otro ejercicio, con su propio cuaderno de resolución y con un proceso de obtención de la solución similar al del problema principal, sólo que más simple. Existe la posibilidad de que el alumno, si conoce la respuesta de este segundo ejercicio, introduzca directamente el resultado sin verse obligado a seguir todo el proceso de resolución que el profesor haya diseñado para el mismo.

Con el fin de comprender mejor el proceso que se produce entre problemas y subproblemas mostramos a continuación la figura 3.2, la cual representa de forma esquemática la interconexión entre distintos tipos de ejercicios en el momento de la ejecución de las acciones de resolución de uno de ellos. Esta representación es común tanto al profesor (durante la resolución que realiza de un ejercicio) como a *MathEdu*, que de esta forma abstrae el proceso de aquel. La referencia de unos problemas a otros la decide el profesor en el momento en que diseña el modo en que ha de resolverse el ejercicio. Como acabamos de explicar, de las dos integrales anteriores, la del coseno y la del seno, la primera de ellas el profesor pide al alumno que la resuelva por sus propios medios y únicamente indique el resultado mediante el correspondiente diálogo de introducción de datos.

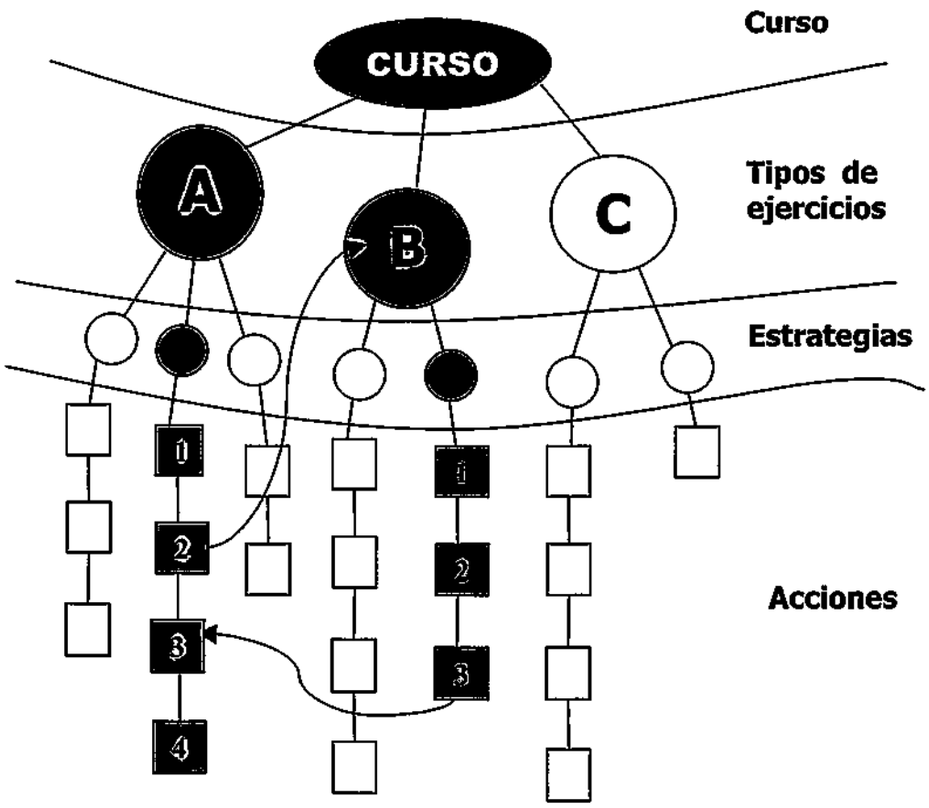

Figura 3.2. Esquema de interrelación entre ejercicios.

Sin embargo para la segunda el profesor ha decidido que debe resolverse como un ejercicio aparte, con su propio cuaderno y su proceso de diálogo con preguntas y respuestas del alumno. El profesor dispone de plena libertad de hacer el diseño como encuentre más oportuno y para ello dispone de los distintos tipos de acciones en *MathEdu Designer* con las cuales puede modelar la resolución según desee.

En la situación hipotética que se muestra en la figura 3.2, el tipo de ejercicio "A" se resuelve utilizando la segunda estrategia del mismo. La segunda acción asociada a esta estrategia requiere que se resuelva un ejercicio de tipo "B" que, a su vez, requiere de otras tres acciones de resolución asociadas a la correspondiente estrategia. Una vez que concluye la resolución del subproblema, continúa la resolución del ejercicio principal con la tercera acción de las especificadas. En Matemáticas esta es una técnica muy común: resolver un problema apoyándose en otros más sencillos que el primero. *MathEdu* no es ajeno a esta circunstancia y posibilita que el diseñador especifique la resolución de un ejercicio basándose en otros más simples. Para esto utilizará un tipo de acción especial que se explicará en el epígrafe siguiente. Este tipo de acción a que nos referimos permite, como decimos, referir partes de un problema a un subproblema de otro tipo, siendo el propio

sistema *MathEdu Solver* el que acepta o no la estrategia que el alumno escoja para resolver el subproblema.

3.2.3. La generalización con *MathEdu Designer*

Hasta aquí hemos hecho una exploración superficial del uso de *MathEdu Solver* en comparación con la resolución mediante lápiz y papel que hemos realizado al inicio. Pero para que *MathEdu Solver* actúe en la manera en que hemos mostrado es necesario un importante proceso de modelado que debe efectuar el profesor en la fase de generalización de ejercicios. El siguiente epígrafe va a profundizar en este proceso, pero antes de explorar con detalle los aspectos relevantes de la herramienta de autor *MathEdu Designer*, queremos aprovechar el ejemplo que acabamos de presentar para mostrar las circunstancias más relevantes de la generalización.

La generalización de un ejercicio pretende obtener un modelo de una clase específica de ejercicios a partir de los datos y características de un ejemplo de dicha clase. Al decir clase de ejercicios nos estamos refiriendo a un tipo concreto (por ejemplo *integración*, *derivación*, *límites*, etc.) y una estrategia de resolución relativa a dicho tipo. En estas condiciones, generalizar un ejemplo requiere de dos tareas principales:

1. La identificación de las *metavariables* del modelo (que se definen a continuación),

2. La definición de las acciones de resolución relativas a la clase de ejercicios que se está modelando.

La primera de las tareas, definición de metavariables, requiere por parte del profesor un proceso de reflexión previo del cual extraiga la decisión de qué elementos del ejercicio va a generalizar. Definimos la metavariable *símbolo* como la terna {*símbolo, condicionante, generador*}. Cada metavariable se instancia cuando se genera un enunciado concreto mediante la asignación de un valor proporcionado por el *generador*. Además, la combinación de los *símbolos* y los *condicionantes* en estructuras simbólicas más complejas adaptadas al lenguaje simbólico de *Mathematica* constituyen patrones de expresiones simbólicas complejas (patrones condicionales), las cuales son usadas para reconocer cualquier expresión matemática que verifique la propiedad expresada por el *condicionante*.

En principio hay infinidad de formas distintas de generalizar un ejemplo y el profesor es el encargado de encontrar la forma más adecuada para que posteriormente, durante el proceso de resolución interactiva, el alumno saque el máximo provecho de los objetivos docentes que el profesor haya marcado para esa clase de ejercicios. Hemos de hacer hincapié en la relevancia del uso de patrones estructurales a los que nos referimos en el capítulo 2. Los patrones estructurales en las fórmulas de un enunciado posibilitan dos tareas importantes

a) la generación de expresiones estructuralmente similares a las del modelo generado por el profesor
b) el reconocimiento de expresiones en ejemplos propuestos por el sistema (*MathEdu Solver)* o por el alumno (con *MathTrainer*).

Por concretar un poco estas ideas consideremos nuevamente el subproblema del ejemplo anterior. En él el integrando es el producto de las funciones

$$u = 5x^2$$
$$v = Cos(x)$$

Ante esta situación el profesor dispone de varias alternativas. Una de ellas puede ser, por ejemplo, generalizar exclusivamente la función trigonométrica, en cuyo caso cualquier otro ejemplo que en el futuro genere *MathEdu Solver* siempre contará con el factor polinómico. También puede hacerlo al revés, generalizando exclusivamente el factor polinómico del integrando y manteniendo fijo el factor trigonométrico (la función coseno en este caso, aunque podría haber sido la función seno igualmente). Imaginemos un tercer caso, en el que el profesor desea generalizar ambos factores y permitir que en futuros ejemplos tomen valores aleatorios. En este caso especificará que:

- El factor polinómico se sustituirá en el futuro por cualquier polinomio de grado menor o igual que 2 con coeficientes enteros menores o iguales que 10 y sin término independiente. Es decir, serán válidas expresiones como

$$x^2 - x$$
$$2x$$
$$-3x^2 + x$$

y no serán válidas expresiones como

$$x^3 - x - 2$$
$$3x^2 + 12x + 5$$

- El factor $Cos(x)$ se sustituirá en el futuro por cualquier expresión trigonométrica simple sobre la variable independiente x, luego serán válidas exclusivamente las funciones $Sen(mx+n)$ ó $Cos(mx+n)$.

Ambas restricciones impuestas sobre los datos del ejercicio darán lugar en el momento de la resolución con *MathEdu Solver* a ejercicios como los que hemos referido más arriba.

El proceso de identificación de valores del ejemplo con variables del sistema que toman valores aleatorios cuando se genera un ejercicio lo denominamos proceso de definición de *metavariables*. Es una de las partes propias de la *programación por demostración* a que hemos hecho referencia en el capítulo 2. Por tanto una *metavariable* constituye la representación generalizada de una parte de los datos del ejercicio utilizado como ejemplo tipo de la clase que está siendo definida. Como se puede comprender, este es uno de los procesos clave en el diseño con *MathEdu Designer*. Más delante profundizaremos en la definición de las metavariables. Por ahora, para comprender de una forma rápida y sencilla los procesos subyacentes a la generalización, nos basta con lo dicho en los párrafos anteriores. A modo de ejemplo podemos indicar que en el caso del ejercicio anterior, las tareas de generalización del mismo que debe realizar el profesor en el último supuesto de los que acabamos de comentar serían equivalentes a:

1. Marcar con el ratón la expresión $5x^2$ en el ejercicio

    a. Declarar el identificador (metavariable) que en el futuro representará las expresiones que sustituyan a $5x^2$. Por ejemplo *polinomio*.

    b. Indicar qué condición se aplicará sobre dichas expresiones. En este caso cualquier expresión que represente la variable *polinomio* debe cumplir la condición de ser, precisamente, un polinomio.

    c. Especificar la función generadora de la metavariable en tiempo de resolución interactiva o guiada. En este caso se debe usar una función generadora de polinomios.

2. Marcar con el ratón el argumento x afectado de la función coseno en el ejercicio

    a. Declarar el identificador (metavariable) que en el futuro representará las expresiones que sustituyan a la x. Por ejemplo *argPolinomial*.

    b. Indicar qué condición se aplicará sobre dichas expresiones. En este caso cualquier expresión que represente la variable *argPolinomial* debe cumplir la condición de ser un polinomio de grado uno.

    c. Especificar la función generadora de la metavariable en tiempo de resolución interactiva o guiada. En este caso se debe usar una función generadora de polinomios.

3. Marcar con el ratón la expresión $Cos(x)$ en el ejercicio

a. Declarar el identificador (metavariable) que en el futuro representará las expresiones que sustituyan al $Cos(x)$. Por ejemplo *trigSimple.*

b. Indicar qué condición se aplicará sobre dichas expresiones. En este caso cualquier expresión que represente la variable *trigSimple* debe cumplir la condición de ser una función trigonométrica simple (seno o coseno).

c. Especificar la función generadora de la metavariable en tiempo de resolución interactiva o guiada. En este caso se debe usar una función generadora de funciones trigonométricas simples.

Tras estas especificaciones efectuadas por el profesor, el ejercicio que ha servido como ejemplo para abstraer su generalización se representará de la siguiente forma en la estructura de datos que manipula *MathEdu* (la cual es transparente tanto para el profesor como para el alumno)

Integrar(*polinomio* · *trigSimple*(*argPolinomial*), x)

Esta expresión representa la generalización de los datos que pueden usarse para resolver cualquier ejercicio de un tipo similar al usado como ejemplo. Del profesor depende que el grado de generalización sea mayor o menor. No es necesario que cambie siempre todos los datos de un ejemplo. En este caso hemos optado por la opción más abstracta posible pero, como decimos, la decisión sobre el grado de generalidad que desee alcanzar depende del profesor exclusivamente.

La segunda de las tareas a que hemos hecho mención anteriormente, la definición de tipos de acciones, también es esencial para dar a la aplicación el enfoque didáctico que el profesor desee. Un ejemplo como

Ejercicio. Calcula la siguiente integral

$$\int x \cdot Sen(x)dx$$

Se resuelve, comúnmente, del modo en que lo hemos hecho con lápiz y papel para la integral I_1. Puede haber ligeras diferencias entre dos personas pero la base conceptual es la misma. Pero si pensamos desde el punto de vista de que es el sistema informático el que va a ir presentando las preguntas diseñadas por el profesor, adaptadas a los datos del ejercicio concreto que se ha generado aleatoriamente, entonces cobra una importancia muy relevante cuáles son las preguntas definidas. Algunos sistemas descritos en capítulos anteriores (*Wiley Web Test*, por ejemplo) hacen uso de un esquema de resolución parecido pero mucho más simple: se le plantea un ejercicio al usuario y este debe introducir la

respuesta (independientemente del grado de dificultad del ejercicio y del tiempo o espacio en papel que haya habido que utilizar para resolverlo). El sistema le dirá al usuario si la respuesta es correcta o no, limitándose a proporcionar la respuesta correcta en el segundo de los casos. Esta situación es absolutamente extrema y *MathEdu* puede funcionar de este modo. Pero qué duda cabe que la verdadera riqueza del sistema está en poder encadenar un proceso de preguntas y respuestas con el alumno que constituyan un diálogo entre el alumno y el sistema.

En estas circunstancias el diseñador debería incluir en su generalización aproximadamente ocho o diez cuestiones para que el alumno fuese contestándolas y, en consecuencia, resolviendo de forma interactiva con el sistema el ejercicio que se le hubiera planteado. Algunas de las preguntas que se pueden extraer de un ejercicio como el anterior son las siguientes:

1. Introduce el valor de u,
2. Introduce v,
3. Calcula la diferencial de u,
4. Calcula V integrando v
5. Selecciona en la paleta la expresión correcta que define una integral por partes
   a. $u \cdot V - \int (V \cdot du) dx$
   b. $u \cdot V + \int (V \cdot du) dx$
   c. $u \cdot V - \int (u \cdot v) dx$
6. La nueva expresión resultante en función de u y v será
7. ...
8. ...
9. Comprueba que la solución que has calculado es correcta: Deriva la *solución*.

En la ilustración 3.6 anterior hemos visto algunas de estas preguntas planteadas al alumno junto con sus correspondientes respuestas. No están contempladas la totalidad de las preguntas pues ocupan un espacio superior al de la imagen. Pensamos que con la ilustración mostrada es suficiente para obtener una idea aproximada del tipo de diálogo que se entabla con el alumno, bien a través de los diálogos ya vistos o bien mediante paletas de opciones como en el caso de la cuestión 5, la cual se muestra en la ilustración 3.7.

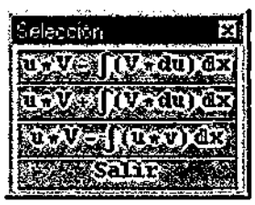

Ilustración 3.7. Paleta de la pregunta 5

Con el ejemplo que acabamos de presentar es posible concluir, para terminar el presente epígrafe y pasar a describir en detalle aspectos propios del diseño de *MathEdu*, que la información relevante contenida en un

ejercicio de cálculo y que es necesario extraer para obtener un modelo del mismo es la siguiente:

a) La clasificación del *tipo de ejercicio* a que corresponde el ejercicio que va a diseñarse.

b) La identificación de los textos y las fórmulas del ejercicio.
   i) Es necesario asignar a cada fórmula del ejercicio un símbolo que la identifique unívocamente.

c) Hay que identificar la *estrategia* a que corresponde el método de resolución del ejercicio.
   i) Esto requiere de la descripción textual de la tarea o tareas en que consiste la estrategia descrita.

d) Hay que definir las *metavariables* contenidas en las fórmulas del ejercicio.
   i) Dar un nombre identificador a cada *metavariable.*
   ii) Establecer una condición que debe cumplir cada *metavariable.*
   iii) Hay que definir una función generadora de cada *metavariable.*

e) Hay que definir las acciones correspondientes a la estrategia de resolución especificada para el tipo de ejercicio.

En el epígrafe siguiente vamos a abordar con profundidad los aspectos relativos a cómo se estructuran en *MathEdu* los datos que acabamos de enumerar, haciendo referencia a las cuestiones propias de la herramienta de autor, la interfaz y las tareas de diseño del profesor.

3.3. Módulo de diseño de ejercicios: *MathEdu Designer*

El módulo que vamos a presentar dentro de este epígrafe constituye la verdadera piedra angular de *MathEdu*, dado que mediante él vamos a modelar y estructurar el conocimiento relativo al planteamiento y resolución de un ejercicio de cálculo simbólico. El trabajo del profesor aquí es esencial ya que el resultado final proporcionado por el módulo de resolución interactiva depende totalmente del trabajo desarrollado en este momento. La herramienta de autor que vamos a describir en este epígrafe ayuda al profesor a seguir una cierta metodología en el análisis y declaración de ejercicios, elaborando con sumo cuidado la información que va a usar para establecer los distintos modelos de ejercicios.

En primer lugar hemos de plantarnos en qué consiste un ejercicio de los que *MathEdu* puede abordar. Cuando nos referimos más arriba a que es capaz de resolver ejercicios que involucren cálculo simbólico no estamos siendo precisos. Pero es que tampoco es posible serlo más, ya que las posibilidades de la herramienta son, en cierto modo, insospechadas ya que es posible concebir la resolución de ejercicios de integración, derivación, límites, ecuaciones diferenciales ordinarias, pero también, por ejemplo, de álgebra lineal, cálculo vectorial, dependencia e independencia lineal, resolución de sistemas de ecuaciones, etc.

Por tanto la tarea de diseño previa que debe realizar el profesor puede ser, en algunos casos, bastante compleja. Evidentemente existen ejercicios en los que la labor de diseño es mínima, pensemos por ejemplo en el modelado de una integral inmediata. Si bien es cierto que algunos casos sí que resultan conceptualmente sencillos de modelar, generalmente (y sobre todo si se pretende un alto grado de generalidad e interactividad), la dificultad en el diseño del modelo es bastante elevada.

En el epígrafe anterior hemos expuesto lo que sería la resolución con lápiz y papel y con *MathEdu* de un ejercicio sencillo de integración por partes. El procedimiento desarrollado para efectuar la resolución no es, en absoluto, trivial. Requiere de determinados conocimientos sin los cuales no es posible ejecutar las diferentes acciones involucradas para alcanzar la solución. *MathEdu* es una herramienta que ayuda al docente en la tarea de explicar a los alumnos qué pasos deben realizarse para resolver los ejercicios. La primera cuestión fundamental es, por tanto, cómo transmitir a *MathEdu* la estructura del ejercicio planteado, las variables relevantes, qué posibles métodos de resolución pueden utilizarse para resolver el ejercicio (estrategias), qué pasos (acciones) han de llevarse a cabo, etc. Vamos a tratar de tender un puente entre dos orillas. A un lado tenemos el lenguaje matemático junto con los distintos aspectos cognitivos que intervienen en la resolución de un ejercicio. Desde la otra orilla vamos a tratar de interactuar mediante un interprete del lenguaje matemático traducido a estructuras de datos, *MathEdu Solver*, que es capaz de establecer con el alumno un diálogo rudimentario (desde el punto de vista de interpretación

de lenguaje simbólico, no natural), pero viable. Y entre medias el profesor debe tender el puente del modelado mediante el uso de *MathEdu Designer.*

MathEdu Designer es un conjunto de procedimientos agrupados en un paquete, que proporcionan la interfaz necesaria para que el profesor defina las estructuras de datos que representan, por una parte, el modelo formal constituido por el texto y las formulas del enunciado y, de otra, los mecanismos de generación de instancias del modelo descrito y las acciones necesarias para dialogar con el alumno y resolver el ejercicio correspondiente a cada instancia.

Ilustración 3.8. Funcionalidad de *MathEdu Designer.*

El acceso al conjunto de procedimientos de *MathEdu Designer* se realiza a través de la paleta que se muestra en la ilustración 3.8. En el diseño de cada nuevo modelo el profesor inicialmente dispone de esta paleta y cada botón de la misma le permite ir definiendo la estructura de datos del curso. En las siguientes secciones vamos a ir describiendo con detalle las tareas que debe realizar el profesor durante el proceso de diseño. Nos limitamos ahora a incluir una breve explicación de las acciones asociadas a cada botón.

1) *Nuevo ejercicio.* Inicia la definición de un ejercicio. Se discrimina por *tipo de ejercicio.*
2) *Generalización.* Permite generalizar los datos del ejercicio que se esta modelando con el fin de poder, en tiempo de resolución, generar y ayudar a resolver ejercicios estructuralmente similares pero con datos distintos.
3) *Acciones.* Contiene el conjunto de acciones de que dispone el diseñador para modelar los métodos de resolución asociados al ejercicio en cuestión.
4) *Guardar el diseño.* Al concluir el proceso de modelado es necesario salvar de forma permanente la estructura de datos creada, con el fin de que pueda utilizarse en sucesivos modelados.

A continuación vamos a iniciar la descripción con detalle del módulo de diseño. Para ello comenzaremos analizando detenidamente cómo se definen ejercicios en Matemáticas, las implicaciones del estilo utilizado y en qué medida afecta esto al diseño con *MathEdu Designer.*

3.3.1. Definición del problema. El lenguaje matemático.

Cuando iniciamos las investigaciones que han conducido a la elaboración de *MathEdu*, y por ende de esta memoria, la primera dificultad con la que nos encontramos fue cómo abordar la estructuración del lenguaje matemático. Las materias científicas y, en particular, las Matemáticas, presentan dificultades inherentes para beneficiarse de los avances más

recientes en las tecnologías educativas basadas en la utilización del ordenador. Estas dificultades son de dos tipos: en primer lugar destacan las dificultades para la construcción de una interfaz de usuario adecuada; en segundo, la construcción de un sistema suficientemente inteligente plantea un reto de gran dificultad. En el primer tipo de dificultades influye el hecho de que las materias científicas, en general, poseen una forma de expresión específica, en la que se combinan habitualmente el texto escrito simple con las fórmulas y con figuras estructuradas, incluyendo referencias explícitas o implícitas entre ellas. En el capítulo anterior hemos puesto de manifiesto cómo el estilo más frecuente en el que se desarrollan textos matemáticos es el semiformal, que combina textos y expresiones formales. Por otra parte, la principal fuente de problemas para la construcción de un sistema que contenga la inteligencia adecuada para la enseñanza de materias científicas es la riqueza y complejidad de los mecanismos de razonamiento subyacentes a las mismas, formalizables a distintos niveles, pero que resultan difícilmente asimilables al nivel al que trabaja un alumno.

La dificultad indicada para el desarrollo de materiales informáticos interactivos para la enseñanza de las Matemáticas y otras ciencias justifica el alto interés que tiene la creación de herramientas de autor específicas para estas materias; diseñadas para ser utilizadas por usuarios expertos, normalmente profesores de la materia a enseñar, que pueden construir sus propios programas interactivos (cursos, colecciones de problemas, etc.) de distintos tipos. Ya hemos comentado anteriormente en 2.6 el uso de técnicas de programación por demostración. Sin embargo, la creación de herramientas en el contexto de aplicaciones para la enseñanza interactiva de materias científicas implica la utilización de interfaces para la manipulación de material de este tipo altamente sofisticadas, lo que hace que aparezcan acrecentados al máximo los mismos problemas citados al comienzo del presente apartado. Pese a todo, las principales ventajas de este enfoque son, en primer lugar, que el trabajo de creación de la herramienta de autor se realiza solamente una vez, reutilizándose posteriormente en las distintas aplicaciones que desarrollan los profesores, y en segundo lugar, que el desarrollo de la interfaz es llevado a cabo por personas altamente cualificadas, capaces de superar las dificultades técnicas inherentes a estos problemas.

MathEdu reduce la complejidad del proceso de desarrollo de las colecciones de problemas. Además, como ya expusimos anteriormente, la flexibilidad y la potencia de la interfaz de la aplicación resultante, en términos del grado de comunicación con el estudiante, permiten una interacción más ágil y unos diálogos más inteligentes. Aunque en realidad los programas desarrollados con *MathEdu* incorporan un tratamiento del conocimiento matemático muy rudimentario desde el punto de vista de la Inteligencia Artificial o de las técnicas más potentes de razonamiento automático, es de destacar cómo el simple incremento de la interactividad y la capacidad de tratamiento simbólico de las aplicaciones, junto con un sistema de reglas de resolución de problemas muy sencillo, permite abordar objetivos hasta ahora difícilmente alcanzables, como el estudio sistemático

de errores comunes entre los estudiantes y la explicación interactiva de sus motivos y de la forma de evitarlos.

Con estas premisas resulta imprescindible comenzar analizando el contenido del enunciado de un ejercicio habitual de Matemáticas. Retomemos uno de los ejercicios considerados más arriba sobre integración

Ejercicio. Calcula la siguiente integral

$$\int x \cdot Sen(x)dx$$

El enunciado presenta, a simple vista, dos partes bien diferenciadas: texto y fórmulas. *MathEdu* no realiza tratamiento alguno de la parte correspondiente al texto. El proceso de diseño involucra las tareas que se realizan sobre las fórmulas. Para poder manipular adecuadamente las fórmulas es necesario disponer de un buen editor de fórmulas matemáticas y ya comentamos en capítulos anteriores que en *MathEdu* utilizamos *Mathematica* como soporte para el desarrollo de la aplicación.

Una fórmula matemática aglutina información conceptual basada en la simbología que emplea. Los signos y símbolos utilizados en una expresión matemática definen el tipo de definiciones, operaciones, o relaciones asociadas a las mismas. En este sentido una expresión como por ejemplo puede ser

$$\lim_{x \to 0}[f(x)]$$

hace referencia, en cualquier contexto, al límite en el origen de la expresión entre corchetes. Desde el punto de vista simbólico no aporta otra información adicional, salvo que para conocer el valor asociado a dicha expresión (caso de existir) es necesario efectuar la operación de obtención del límite de una función en un punto concreto. Sin embargo, si contextualizamos el valor de la operación anterior entonces sí que conlleva información adicional. Si consideramos por ejemplo que la función

$$f(x) = \frac{x \cdot Sen(x)}{1 - Cos(x)}$$

entonces el valor del límite es 2, lo cual indica, como primer dato relevante, que el límite existe.

Ahora bien, la información contenida en una fórmula no es sólo la que aparece de forma explícita a través de los operadores, signos y símbolos. Un conocimiento profundo de la teoría de las Matemáticas aporta datos implícitos contenidos en una expresión

matemática. Este conocimiento de la materia en ocasiones es suficiente para que la persona que manipula una fórmula sepa cómo debe manejarla. Por ejemplo, el ejercicio

Ejercicio. Calcula la siguiente integral

$$\int Sen(x)dx$$

por su propia estructura pone de manifiesto que para ser calculada debe resolverse como una integral inmediata. En ocasiones no es tan evidente esta información implícita y es necesario valerse de algún cálculo previo para determinar qué método de resolución debe emplearse. Por ejemplo, el ejercicio

Ejercicio. Calcula el valor de

$$\lim_{x\to\infty}\left[\frac{x^3-2x+2}{x^3+x}\right]^{x^2}$$

debe resolverse utilizando la definición del número *e*, pero para llegar a determinar el método hemos tenido que efectuar una resolución parcial previa llegando a la indeterminación 1^{∞}, la cual nos invita al método ya indicado.

Por supuesto que existen ocasiones en las que un enunciado de un ejercicio puede especificarse utilizando bien texto en lenguaje natural o bien notación matemática formal pura. Para ilustrar esta última situación planteada pensemos por ejemplo en el ejercicio

Ejercicio. Calcula la derivada de la función

$$y=\frac{x^2-2x+1}{e^{-x}}$$

Este mismo ejercicio puede plantearse en términos estrictamente formales como

Ejercicio. Calcula

$$y'=\partial_x\left(\frac{x^2-2x+1}{e^{-x}}\right)$$

Comprobamos, por tanto, que el lenguaje semiformal matemático denota distintos niveles de información que puede aparecer, bien en forma de texto, en lenguaje natural, bien en forma de operadores y símbolos con un significado preciso. El modo en que estas

consideraciones sobre el estilo en que están formulados los ejercicios van a influir sobre el proceso de diseño parece bastante evidente. El profesor va a ser el encargado de redactar los ejercicios y, tomados como ejemplo, los generalizará para que puedan servir de soporte a la generación de otros ejercicios nuevos. Qué duda cabe que es muy importante el modo en que el profesor plantee los enunciados de los ejercicios dado que *MathEdu* sólo va a permitir actuar sobre las expresiones simbólicas que constituyen las fórmulas, en ningún caso (al menos en la versión actual del sistema) sobre textos en lenguaje natural. En consonancia con esto, una de las tareas principales que debe realizar el diseñador es establecer una jerarquía con los distintos casos que pueden presentarse para un mismo tipo de ejercicio. Esta jerarquía se equipara con un *Curso* en la estructura de datos de *MathEdu*. Para diferenciar tipos de ejercicio no son relevantes únicamente las expresiones concretas que aparezcan en una fórmula. Es, si cabe, más importante la forma en que se debe enfocar la resolución del ejercicio a partir de los datos del mismo. Conviene recordar que ejercicios con datos distintos se pueden resolver de idéntica forma. Por ejemplo, desde el punto de vista de la resolución de una integral, las expresiones

$$\int x \cdot Sen(x)dx$$

y

$$\int x \cdot Cos(x)dx$$

son equivalentes. El hecho realmente relevante es que en ambos casos el integrando se compone de un producto de dos funciones, una polinómica y otra trigonométrica simple. Ambas se resuelven de igual modo. El profesor debe abstraer este hecho y, valiéndose de las herramientas que le proporciona *MathEdu Designer*, modelar ambos ejercicios como un sólo ejercicio de *integración* cuya resolución se realiza mediante la *estrategia* de integración *por partes*.

Llegados a este punto parece necesario concretar con mayor precisión aspectos propios de la representación formal en el lenguaje de *MathEdu*. En las siguientes secciones vamos a detallar las características del mismo, aunque constantemente recurriremos a algunos de los ejemplos planteados anteriormente para hacer más clara la exposición.

3.3.2. Abstracción de datos.

En el punto anterior hemos comenzado a introducir aspectos propios de la modelación formal de un ejercicio en el lenguaje propio de *MathEdu*, como son el tipo de un ejercicio (por ejemplo *integración*) o una estrategia de resolución de ejercicios de integración (*por partes*).

En el epígrafe 3.2.3 introdujimos el concepto de *metavariable* utilizando un ejemplo y definiendo el concepto a partir de dicho ejemplo. En el presente apartado vamos a detallar con más exactitud qué son y cuál es el papel que juegan las metavariables en el proceso de diseño de ejercicios con *MathEdu Designer*.

Ya vimos que la definición de una metavariable involucraba datos relativos al *símbolo* asignado, el predicado o procedimiento *condicionante* y el procedimiento *generador*. En el ejemplo descrito en 3.2 quedó de manifiesto que este tipo de estructuras son necesarias para la generalización de ejercicios y su posterior generación aleatoria y presentación al alumno. Posteriormente analizaremos con detalle mediante ejemplos cómo se programan las metavariables.

La estructura de datos de un curso de *MathEdu* se organiza como un conjunto de datos en el que se representan los distintos *tipos de ejercicios* junto con información relativa a su enunciado, la estrategia o estrategias que se pueden usar para los ejercicios de ese mismo tipo, las metavariables involucradas (junto con todos los aspectos relativos a su generación)

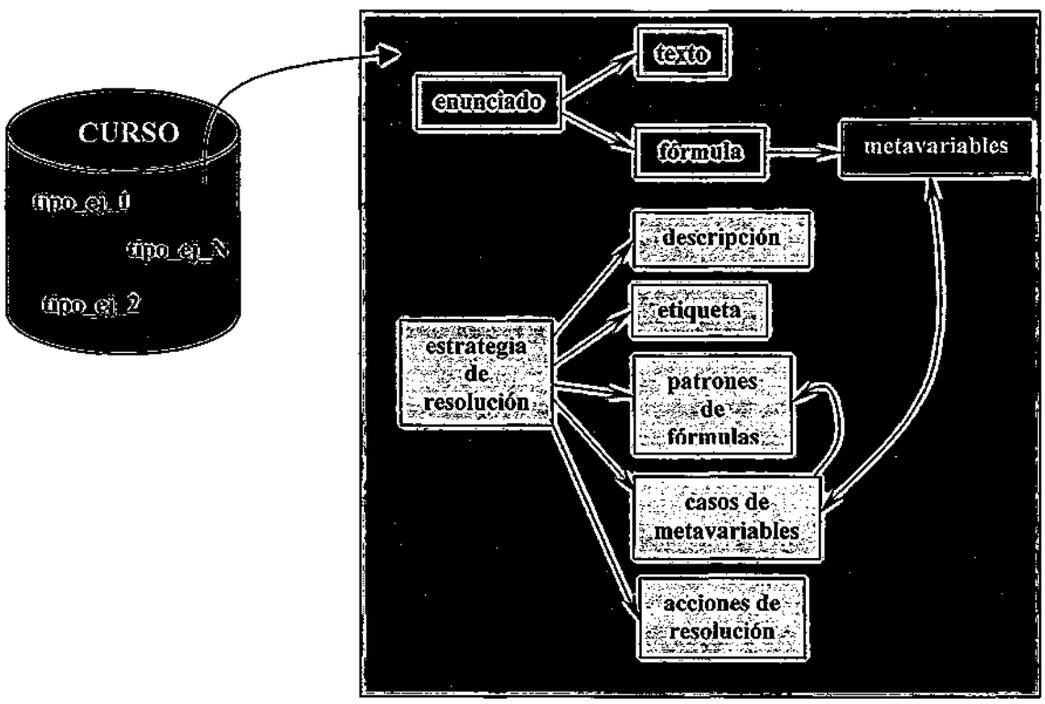

Figura 3.3. Descripción abstracta de los ejercicios de un curso.

y las acciones necesarias para la resolución de los ejercicios según las distintas estrategias. En el apéndice A (apartado A) se muestra la organización de la estructura formal de los datos en un curso de *MathEdu*. Es destacable la sencillez de la abstracción jerárquica de los

datos de un ejercicio, sin que ello represente una merma desde el punto de vista de la representación y gestión del conocimiento propio de los cursos.

En la figura 3.3 mostrada se representa la estructura de un ejercicio desde un punto de vista global. Cada curso está formado por una colección de ejercicios cada uno de los cuales se compone, a su vez, de un enunciado y de una o más estrategias de resolución. Así, por ejemplo, un curso de Cálculo involucrará ejercicios de *integración*, *derivación* y *límites*. El tipo *integración* puede comprender ejercicios de integración de funciones exponenciales, logarítmicas, potenciales, integración de funciones trigonométricas, por partes, racionales, etc. Cada uno de estos tipos tiene en común con todos los demás en que la estructura del enunciado es similar en todos los casos: se debe calcular la integral de una función que depende de unas metavariables. Dependiendo de la función que se trate se debe emplear una estrategia u otra para la resolución. Las estrategias describen la forma en que debe resolverse el ejercicio planteado en el enunciado. Para ello requieren la descripción de los patrones de las fórmulas del enunciado y las acciones de resolución. Las acciones son genéricas para todo ejercicio que contemple un mismo enunciado con distintos casos de metavariables. Por ejemplo se resuelven del mismo modo las integrales

$$\int x \cdot Sen(x)dx \quad \text{y} \quad \int x \cdot ln(x)dx$$

a pesar de la diferencia entre los integrandos. Estructuralmente ambas expresiones son equivalentes. Condicionalmente no lo son. La primera tiene un factor con el condicionante de ser función trigonométrica y la segunda un factor con la condición de ser función logarítmica. Estas dos integrales son dos casos distintos de una misma estrategia. En resumen, por casos de metavariables estamos refiriéndonos a la situación en la que, basándonos en los patrones estructurales y condicionales asociados a una fórmula dependiente de un conjunto de metavariables, es posible generar y reconocer expresiones formadas por distintas funciones, manteniéndose en cualquier caso la uniformidad de la estrategia asociada. Más adelante volveremos a insistir en cuestiones relacionadas con los casos de un mismo ejercicio.

MathEdu Designer, en su propósito de herramienta de autor, proporciona al diseñador del ejercicio la interfaz adecuada para definir los datos que representan el modelo de un ejercicio. Este proceso de definición requiere de una serie de tareas básicas que enumeramos a continuación:

a. Definir el enunciado como un conjunto de texto y fórmulas.
   - Definir las fórmulas e identificarlas con los símbolos que las representarán.

b. Asignar a los símbolos un patrón (que quedará al final del proceso en función de las metavariables) que defina el contenido de la fórmula. Este proceso se realiza durante la generalización.

c. Definir una estrategia de resolución para el tipo de ejercicio definido.
   - Declarar las metavariables.
   - Especificar las acciones de resolución asociadas a dicha estrategia.

Cada uno de estos procesos enumerados va a ser descrito con detalle en la sección siguiente hasta el final del presente epígrafe.

3.3.3. El modelo del ejercicio.

Comenzamos en la presente sección la descripción del proceso de definición de datos. En primer lugar declararemos un tipo de ejercicio para, a partir de él, poder introducir en el sistema distintas estrategias de resolución para dicho tipo. Con ello detallaremos simultáneamente la estructura de los datos que representan a los ejercicios y la forma en que el profesor los especifica.

3.3.3.1. Tipo de ejercicio.

Vamos a describir cómo se va realizando la identificación entre conceptos extraídos del ejercicio y la estructura de datos asociada al mismo. Bajo una misma denominación de tipo de problema (por ejemplo, EDO, integración o derivación), pueden agruparse multitud de ejercicios distintos, con una característica común que los aglutina, generalmente la estructura de sus fórmulas. La clasificación de los distintos tipos de ejercicios se realizará basándose en las distintas estrategias de resolución válidas para un mismo tipo. Distinguiremos entre distintas estrategias más adelante.

Cuando el diseñador desea declarar un nuevo tipo de ejercicio, *MathEdu Designer* le proporcionará al profesor la interfaz necesaria para declarar un **identificador** de dicho tipo así como el **enunciado** que dicho tipo conlleva asociado. Como ya se comentó en el apartado anterior, el enunciado se compondrá de una parte de texto, en lenguaje natural, la cual no se procesa en ningún momento, y una o más fórmulas relativas al enunciado del ejercicio. En definitiva será el ejercicio tal y como el profesor lo desea escribir en el *cuaderno de diseño*. Cada fórmula debe ser identificada mediante un símbolo. Este símbolo va a ser utilizado para abstraer el enunciado y poder referirse a las fórmulas generalizadas del ejercicio haciendo uso del mismo. Pongamos un ejemplo. Supongamos que el diseñador ha definido el tipo de ejercicio *derivación* e introduce un enunciado para describir estos ejercicios. Este puede ser

"Obtén la derivada de la expresión 2x +1"

Esta expresión incluye una fórmula que el profesor debe identificar, la interfaz de *MathEdu Designer* le permite hacer esto simplemente marcándola con el ratón, debiéndole asignar un símbolo identificador. Por ejemplo supongamos que define

$$\textit{función} \rightarrow 2x+1$$

En consecuencia, el enunciado del tipo de ejercicio *derivación* se transforma en

Enunciado("Obtén la derivada de la siguiente expresión", *función*)

El símbolo *función* se describirá posteriormente en términos de las metavariables que especifique el profesor y en cada ejecución de *MathEdu Solver* se sustituirá dicho símbolo por los ejemplos que se generen oportunamente a partir de las metavariables definidas. Por el momento quedémonos con el hecho relevante de que el enunciado se transforma en una representación abstracta del mismo en el que las fórmulas se sustituyen por sus representaciones simbólicas.

La ilustración 3.9 muestra el diálogo de definición de un enunciado de tipo *integración*. A partir de este diálogo se pueden declarar los símbolos relativos a las fórmulas del mismo o bien, una vez que han sido declaradas todas las fórmulas, crear la estructura del ejercicio subyacente para este enunciado.

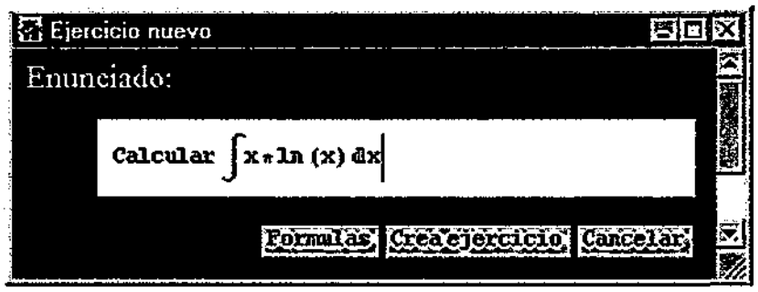

Ilustración 3.9. Diálogo de definición del enunciado.

Para cada una de las fórmulas en el enunciado el diseñador debe asignarles un símbolo identificador. En este punto es necesario destacar que la fórmula queda ligada a dicho símbolo en términos de la expresión simbólica representada en el lenguaje simbólico propio de *Mathematica*. Esto es necesario porque cuando posteriormente el diseñador generalice la fórmula, identificando partes de la misma con metavariables del problema, el símbolo definido aquí hará referencia a la fórmula en función de dichas metavariables. Dicho en otros términos, la fórmula tal y como aquí aparece, se va convertir en un patrón estructural de expresiones similares, en las que las únicas variaciones que van a darse son los datos que aparezcan en la misma, no la estructura subyacente, la cual siempre estará ligada al identificador simbólico que se le ha asignado.

Desde el punto de vista de la estructura simbólica de los datos en *MathEdu*, es importante indicar que con los datos introducidos a través del diálogo correspondiente para la asignación de identificadores simbólicos a fórmulas, se define la estructura simbólica siguiente que permitirá, cuando sea necesario, obtener el patrón estructural de la fórmula a través de su símbolo (en este caso suponemos que el profesor lo ha denominado **integral**)

$$\text{patrones\_de\_formula}(\textbf{integral} \rightarrow \int (x \cdot \ln x) dx)$$

La interpretación de esta expresión, en términos más coloquiales, establece que la fórmula referenciada mediante el símbolo **integral** tiene la estructura descrita en la parte derecha de la asignación. Durante el proceso de generalización de ejercicios se va a transformar la expresión de la integral utilizada por el diseñador como ejemplo en un patrón para generar expresiones equivalentes a esta. Esto se conseguirá mediante el uso de metavariables. El paso desde la expresión simbólica de la fórmula a su correspondiente patrón simbólico en función de las metavariables no es inmediato. El diseñador acaba de iniciar el proceso de definición de un ejercicio y aún no han sido definidas las metavariables que va a necesitar y, por tanto, es imposible asignar un patrón simbólico a la fórmula. Por el momento sólo se toma nota de cuáles son las fórmulas y su posición en el enunciado.

En cuanto a la estructura de datos asociada al ejercicio generada hasta el momento por el profesor, se resume en los elementos de la tabla 3.2

definición	
tipo de ejercicio	**integración**
enunciado	Cuaderno("Calcular", **integral**)
patrones de fórmula	**integral** = Integrar(x·ln(x), x)

Tabla 3.2. Ejemplo de definición de datos de un ejercicio.

3.3.3.2. Estrategias.

Antes de definir formalmente las estrategias en *MathEdu* vamos a explicar qué queremos representar con dicho concepto. La idea de estrategia surge de la de sistematicidad en la resolución de ejercicios. Es habitual que al querer resolver algún ejercicio que involucre cálculo simbólico pensemos en algún *método de resolución* asociado al tipo de ejercicio planteado. Así, por ejemplo, hablamos de resolución de integrales por cambio de variable, por partes, o por descomposición en fracciones simples. En el caso de EDO, hablamos de resolución de ecuaciones homogéneas, exactas, separables, etc. Y si se

trata de obtener la derivada de una expresión solemos derivar productos, cocientes, aplicamos la regla de la cadena, etc. Para cada tipo de ejercicio aparentemente existen distintos métodos apropiados de resolución. La idea de estrategia esta relacionada con esta circunstancia. Vamos a considerar una estrategia como un método de resolución válido para un tipo de ejercicio. Como cada ejercicio puede agrupar datos muy diferentes, existirán distintas estrategias asociadas a cada tipo (ver apartado A del Apéndice A).

Una ***estrategia*** en *MathEdu* es una estructura simbólica (tipo abstracto de dato) representada por un símbolo identificador y descrita según los campos que a continuación se detallan,

- una ***descripción textual*** del significado de la estrategia,

- una ***descripción formal*** de la información relativa a los patrones de las fórmulas contenidas en la definición,

  Ejemplo

  Para representar la fórmula $\int x \cdot \ln(x) \cdot dx$ del enunciado anterior se identifica mediante el símbolo *integral* y se generaliza mediante el patrón

  Integrar(*polinomio* · *funcion*(x), x)

  en el cual la expresión x aparece representada por la metavariable *polinomio*. Por tanto la descripción formal de los patrones de las fórmulas asociadas a la estrategia que se esté definiendo será

  patrones_de_formula(**integral** → Integrar(*polinomio* · *funcion*(x), x))

- un conjunto de *listas de especificación de metavariables*, que representan los distintos casos de funciones que pueden aparecer en las fórmulas del enunciado,

  Ejemplo

  En el patrón de expresiones Integrar(*polinomio* · *funcion*(x), x) es posible identificar las metavariables *polinomio* y *funcion* con distintas funciones generadoras del mismo, de forma que en el momento de generar ejercicios para presentárselos al alumno se utilicen indistintamente unas funciones u otras. En este ejemplo en concreto se puede distinguir entre polinomios con o sin término independiente o bien que *funcion* represente las funciones seno, coseno o logaritmo y que haya, en consecuencia, diferentes casos asociados a un mismo patrón (ver apartado B del Apéndice A).

- un conjunto de *acciones de resolución*, las cuales detallan el procedimiento a seguir para resolver los ejercicios descritos en el enunciado.

Además de estos elementos descritos, una estrategia lleva asociados un **símbolo identificador** a efectos de manipulación del sistema y una **etiqueta** (para utilizarla en las paletas de estrategias).

Para completar la descripción anterior de la estructura de datos que corresponde a una estrategia es necesario describir cada uno de los componentes indicados (metavariables y acciones), lo cual se explicará en las secciones 3.3.3.2.1 y 3.3.3.2.2, correspondientes a las fases de generalización y definición de acciones. Como se puede deducir de la definición de estrategia que acabamos de detallar, el concepto de estrategia es complejo, su elaboración requiere distintas acciones por parte del diseñador, utilizando para ello distintas interfaces. Desde el punto de vista del tiempo que emplea el diseñador en definir un ejercicio, declarar y construir una estrategia es la actividad de diseño más extensa.

Una vez que se han especificado los datos básicos para la identificación de la estrategia que se desea definir (identificador, etiqueta y descripción), el sistema muestra, por vez primera desde el inicio del diseño del ejercicio, el cuaderno de diseño. Inicialmente sólo aparece descrita la información relativa al enunciado, especificando por separado la parte correspondiente a texto y la relativa a fórmulas. La ilustración 3.10 muestra un ejemplo del cuaderno de diseño correspondiente al ejercicio de integración definido en la ilustración 3.9, en el momento en que ha realizado las siguientes actividades

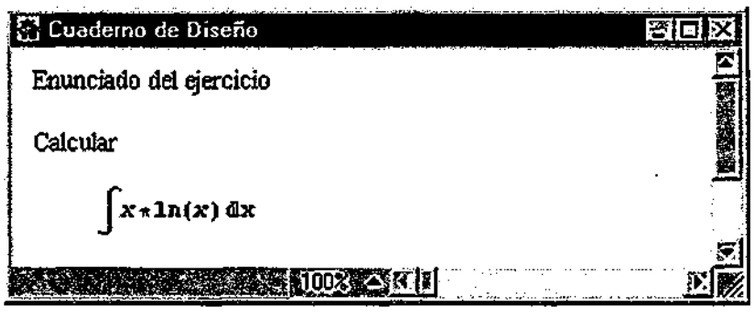

Ilustración 3.10. Enunciado en el *Cuaderno de Diseño*.

i. ha declarado un tipo de ejercicio nuevo,
ii. ha introducido el enunciado de un ejercicio del tipo declarado,
iii. ha identificado las fórmulas del enunciado, asignándoles un símbolo identificador,
iv. ha declarado una estrategia para resolver el ejercicio que ha planteado, asociándole a dicha estrategia un símbolo identificador y un texto de descripción de la misma.

A partir de este momento el profesor debe proceder a efectuar la generalización del ejercicio y especificar las acciones de resolución. Las tareas correspondientes a dichas actividades básicas del diseño las describiremos en las siguientes secciones.

3.3.3.2.1. Generalización del problema: Programación basada en ejemplos.

Una vez que el profesor ha especificado los datos iniciales del ejercicio, como son, el tipo al que pertenece, el enunciado y la estrategia que va a describir las acciones de resolución básicas para el mismo, el siguiente paso fundamental en el proceso de diseño corresponde a la ***generalización*** del ejercicio. El objetivo de este proceso es convertir el enunciado particular introducido por el profesor al inicio del diseño en un enunciado general, válido para cualquier ejercicio equivalente al ejemplo. Es decir, en términos más concretos, se pretende convertir la fórmula del enunciado

Ejercicio. Calcula la siguiente integral

$$\int x \cdot Sen(x)dx$$

en el patrón

$$\int (u\_ \cdot v\_)dx$$

siendo u y v expresiones que generalizan el caso particular de x y $Sen(x)$ respectivamente.

Por consiguiente, entendemos por ***generalización*** el conjunto de tareas del proceso de diseño mediante las cuales declaramos las metavariables, elementos que definen los patrones de las fórmulas del enunciado.

Antes de proceder a detallar de qué forma se realizan estas tareas, vamos a exponer algunos ejemplos al respecto de este proceso. Consideremos nuevamente el ejercicio del epígrafe 3.2.

EJEMPLO 1

Consideremos el siguiente ejercicio.

Ejercicio. Calcula la siguiente integral

$$\int 5x^2 \cdot Cos(x)dx$$

Puesto que ya se detalló en 3.2 la forma de resolver este ejercicio, vamos ahora a incidir en las consecuencias que, desde el punto de vista de *MathEdu*, tiene el proceso de generalización. La generalización de un ejercicio es una actividad que realiza el profesor con

el fin de que el ejercicio sobre el que efectúa el diseño sirva de ejemplo para la generación aleatoria de otros ejercicios similares cuyo método de resolución sea el mismo en todos ellos. El efecto que se consigue es similar al de crear una base de datos de ejercicios asociados a los métodos de resolución de todos ellos (utilizadas, por ejemplo, para facilitar a los profesores la generación aleatoria de exámenes).

Si analizamos con detenimiento el ejercicio anterior veremos que la información concerniente a la fórmula que figura en el mismo hace referencia a

- el operador integral,
- el factor polinómico $5x^2$,
- el factor trigonométrico $Cos(x)$.

Si abstraemos esta información contenida en la expresión y sustituimos

- x por un polinomio $P(x)$ de cualquier grado,
- $Sen(x)$ por $Cos(x)$ o por a^{bx}

El método de resolución (la estrategia) no varía y, sin embargo, tenemos acceso a una cantidad notable de ejercicios distintos. El objetivo, como acabamos de ver es, por tanto, sustituir en el ejercicio la expresión de la fórmula

$$\int 5x^2 \cdot Cos(x)dx \qquad [1]$$

por el patrón

$$\text{Integrar}(u\_ \cdot v\_, x) \qquad [2]$$

La expresión [2] difiere de la [1] en que se han sustituido las funciones originales del integrando por los símbolos identificadores de las metavariables ($u\_$ y $v\_$). Como recordaremos de la definición de metavariable dada en el apartado 3.2.3, una metavariable está formada por la terna {*identificador, condicionante, generador*}. La expresión [2] establece la definición de la fórmula cuyo símbolo es *integral*, en función de las metavariables

$$\{u, PolynomialQ, polyGen(2, 0)\} \qquad [3]$$

y

$$\{v, TrigQ, simpleTrigGenerator(any)(x)\} \qquad [4]$$

La expresión [3] especifica que el símbolo u en la fórmula [2] debe ser reemplazado, cuando se genere un ejercicio, por el polinomio generado con la función generadora de polinomios hasta grado 2, con coeficientes aleatorios enteros y sin término independiente. Por otra parte, el predicado condicionante del símbolo u es *PolynomialQ*. Este es el predicado que se ha utilizado en tiempo de resolución y al que ya hicimos referencia en el epígrafe 3.2. En el epígrafe 2.5.2 se explicó cómo se pueden efectuar operaciones booleanas sobre patrones utilizando la función **MatchQ** y ya vimos entonces como se puede aplicar dicha función junto con los patrones condicionales para la verificación de fórmulas.

Tal y como acabamos de explicar para la expresión [3], la expresión [4] especifica que el símbolo v en la fórmula [2] debe ser reemplazado, al generar un ejemplo, por cualquier función trigonométrica simple generada aleatoriamente con la función generadora *simpleTrigGenerator(any)*. Por otra parte, el predicado condicionante del símbolo v es *TrigQ*. Puesto que *Mathematica* proporciona únicamente un predicado relativo a propiedades de funciones (*PolynomialQ*), para facilitar la labor de los diseñadores hemos definido un paquete con otros predicados habituales. Evidentemente la lista no es en absoluto exhaustiva y si el profesor desea utilizar algún predicado no definido debería ser él el que lo programase.

EJEMPLO 2

Veamos otro ejemplo desde una perspectiva más próxima a las estructuras de datos subyacentes en el mismo. El ejemplo que vamos a presentar a continuación hace referencia a un aspecto que ya ha sido tratado en el apartado 3.3.2 cuando nos referimos a los distintos casos que pueden presentarse para un mismo enunciado. Consideremos la expresión,

$$\int x \cdot e^{2x} dx \qquad [5]$$

que como podemos comprobar difiere de [1] en la función exponencial que multiplica a la expresión polinómica x. Desde el punto de vista del modelado de este nuevo ejercicio existen evidentes diferencias respecto de aquel, siendo el patrón que generaliza [5]

$$\text{Integrar}(u\_ \cdot v\_ , x) \qquad [6]$$

el cual coincide con [2], pero las metavariables definidas involucran un patrón condicionado diferente

$$\{u, PolynomialQ, polyGen[2, 0]\} \qquad [7]$$

y

$$\{v, ExpQ, e^{polyGen[1,0]}\} \qquad [8]$$

Sin embargo, a pesar de los cambios realizados en el ejercicio y de las consiguientes modificaciones en el modelo, los métodos de resolución son similares en ambos casos. Y esto es lo realmente relevante. Es decir, las mismas acciones que deben ejecutarse para resolver el ejercicio [1] deben ejecutarse para resolver el [5].

En resumen, a partir del enunciado del ejercicio el profesor debe ser capaz de construir expresiones simbólicas que generalicen el mismo. Para realizar la generalización dispone de herramientas de ayuda, tanto desde el punto de vista de la interfaz (la cual describiremos a continuación) como desde el punto de vista de las funciones y predicados de que dispone al efecto.

Pero antes de explicar en qué consiste la interfaz propia de la generalización de ejercicios, es importante destacar el siguiente hecho relevante desde el punto de vista de la estructura de datos resultante tras el proceso de generalización. La estructura subyacente a las fórmulas de un ejercicio es común para todos los ejercicios concebidos bajo un mismo tipo. Así, todos los ejercicios de tipo *integración*, por ejemplo, tienen en común que la estructura de la fórmula que representan viene dada por el patrón

$$\text{Integrar}(expr\_, x) \qquad [5]$$

siendo *expr_* cualquier expresión matemática. Dependiendo de la expresión que se modele se dará lugar a ejercicios con métodos de resolución (estrategias) completamente distintos. Por ejemplo, la integral

$$\int \frac{x+1}{x^2+3x-2}dx$$

se resuelve como una integral racional, mientras que

$$\int \frac{Sen(2x)}{1+Sen(2x)}dx$$

puede resolverse haciendo el cambio de variable $u = tg(x)$.

Esta particularidad no debe escapar al diseño de la estructura de datos subyacente en el curso diseñado con *MathEdu* y, aunque el profesor tiene potestad para realizar el diseño como quiera, parece lógico pensar que si es posible ahorrar trabajo es preferible hacerlo. Desde un punto de vista pedagógico el alumno, al resolver con los mismos métodos

ejercicios aparentemente diferentes por los datos que contienen, percibe y puede abstraer el concepto de estrategia. Por tanto es imprescindible especificar, dentro de una misma estrategia de resolución, los distintos casos a que dan lugar los modelos con metavariables que proporcionan distintos valores y/o condicionantes. Tal y como vimos en el ejemplo del apartado 3.3.2. se contempla la posibilidad del modelado de los casos de una estrategia en la estructura de datos del curso (descrita en el apartado B del apéndice A). Si recordamos el ejemplo a que nos referimos de las integrales

$$\int x \cdot Sen(x)dx \quad \text{y} \quad \int x \cdot ln(x)dx$$

la forma en que se efectúa el modelado es mediante una lista formada por distintas especificaciones de metavariables. De este modo estamos contemplando la posibilidad de que distintos tipos de ejercicios, con el factor común de que se resuelven del mismo modo, se agrupan bajo el "paraguas" de una misma estrategia; con la ventaja evidente de que el diseño de las acciones pertinentes para la resolución sólo es necesario definirlas cuando se establece por primera vez la estrategia en cuestión (*por partes*, en el caso del ejemplo que nos ocupa). La representación de la estructura resultante para los casos [1] y [5] será

Estrategia: Por partes

casos

lista 1 de especificación de metavariables
u: polinomios de grado 2 sin término independiente,
v: función trigonométrica simple.

lista 2 de especificación de metavariables
u: polinomios de grado 2 sin término independiente,
v: función exponencial con exponente polinómico de grado 1 sin término independiente

Una vez que hemos expuesto un par de ejemplos relativos a cuestiones propias de la generalización al objeto de aclarar, de manera informal, aspectos relativos a conceptos relevantes de la misma, vamos a profundizar en la interfaz que proporciona *MathEdu Designer* al diseñador para llevar a cabo su tarea de modelado.

La interfaz de generalización de ejercicios

La forma en que se generaliza el ejercicio es un claro ejemplo del paradigma de programación mediante ejemplos. A partir del enunciado introducido por el profesor y mediante un proceso interactivo con una serie de tareas estándar, se transforma el contenido del ejercicio en conocimiento implícito que representa la generalización contenida en las metavariables. El mecanismo de grabación de la descripción de datos se inicia al seleccionar la opción de generalización del menú principal de *MathEdu Designer* y

concluye al cerrar el diálogo de generalización que pone fin a la definición de metavariables de un ejercicio. Este procedimiento de programación mediante ejemplos, descrito en (Cypher, 93) consiste en almacenar en forma de comandos del sistema las tareas necesarias para componer una tarea más compleja, tal y como vamos a explicar a continuación.

En las secciones y párrafos anteriores hemos expuesto diversos ejemplos acerca de los procesos que tienen lugar y sobre los conceptos que subyacen en la generalización de un ejercicio. A continuación vamos a ilustrar el proceso de generalización haciendo uso de los ejemplos anteriores. Para fijar el contexto en el que se van a desarrollar las tareas que se van a realizar recordaremos la situación de partida. El profesor ha concluido la fase de declaración del enunciado del ejercicio la cual llevaba aparejada la especificación de

- tipo de ejercicio
- estrategia de resolución
- nombres simbólicos para las fórmulas del ejercicio

I. Generalización de ejercicios nuevos

El proceso de generalización se realiza mediante el diálogo correspondiente a la ilustración 3.11. Como puede comprobarse está compuesto de cuatro campos para introducción de datos y cuatro botones. El proceso de definición de metavariables requiere seleccionar con el ratón las subexpresiones de la fórmula que se deseen generalizar y, a través del diálogo mostrado, proceder a su generalización. Este es un proceso secuencial y debe realizarse tantas veces como metavariables se desee definir.

La descripción de los elementos relevantes del diálogo es la siguiente:

- *Nombre*: campo para introducir el nombre de la metavariable que se define
- *Valor*: campo que contiene el valor correspondiente de la metavariable en el ejercicio que sirve de ejemplo para la definición de la misma.
- *Condición*: condición que especifica el diseñador y que deben cumplir las expresiones que desee que verifiquen la coincidencia de patrones con la metavariable definida.
- *Generador*: función generadora de valores aleatorios para la metavariable definida.

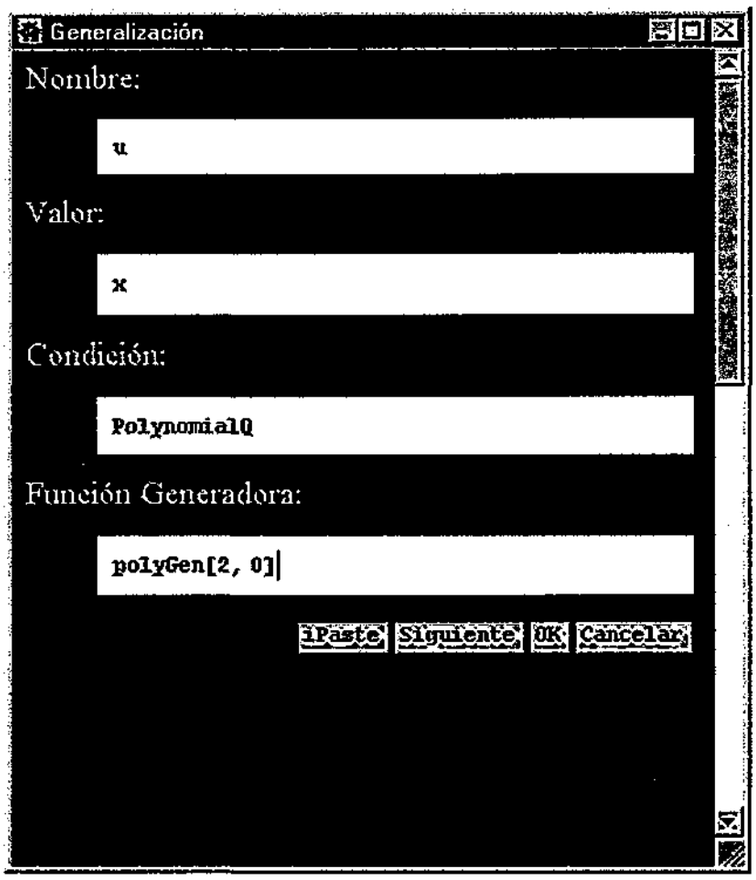

Ilustración 3.11. Diálogo de generalización.

En cuanto a la funcionalidad asociada a los botones, hemos de destacar los siguientes

- *iPaste*: botón que se utiliza para efectuar las operaciones *copiar/pegar*. Estas operaciones no se efectúan como meras acciones de edición, sino que involucran un tratamiento de datos más complejo transparente al diseñador. Explicaremos más delante en qué consiste dicho tratamiento (llamémosle de manera informal "pegado inteligente").

- *Siguiente*: botón que vacía el contenido de los campos de edición para poder volver a utilizarlos en la definición de la siguiente metavariable. Cuando la metavariable que está siendo definida es la última, al concluir el proceso se debe pulsar *OK*.

Las operaciones que se realizan con el botón *iPaste* una vez que el diseñador ha marcado con el ratón en la fórmula del cuaderno de diseño la subexpresión que desea generalizar son las que detallamos a continuación:

- Para la subexpresión marcada el sistema establece delimitadores de dicha expresión en el conjunto de la fórmula.

- Se lee toda la fórmula con las marcas incluidas
- Se determina la posición de las marcas en la fórmula leída
- Se obtiene el identificador simbólico de la fórmula
- Se restablece la fórmula a su forma original
- Se copia en el diálogo de generalización la subexpresión marcada con el ratón por el diseñador.

En el proceso de pegado inteligente se almacena la posición que ocupa, dentro de la fórmula original, la subexpresión marcada por el profesor. Este proceso es transparente para el diseñador y es de suma trascendencia porque, gracias al mismo, se generan los patrones estructurales que permiten reconocer cuándo se aplica una estrategia determinada, de modo que en futuros ejemplos extraídos a partir de la fórmula original se dispondrá de la información suficiente sobre la estructura de la expresión simbólica a generar.

Sobre la cuestión de cuáles son las metavariables que han de definirse, esa es una consideración que debe analizar el profesor basándose en los objetivos que persiga con el ejercicio. Un mismo ejercicio admite múltiples generalizaciones diferentes. Obviamente, la opción que el diseñador escoja es potestad suya. Un ejemplo de esta situación puede ser el siguiente.

La expresión [5] mostrada anteriormente

$$\text{Integrar}(x e^{2x}, x)$$

es posible modelarla como hicimos en [6] mediante el patrón estructural

$$\text{Integrar}(u\_ \cdot v\_, x)$$

o bien como

$$\text{Integrar}(x \cdot v\_, x) \qquad [9]$$

siendo en ambos casos

$$\{v, \mathit{PolynomialQ}, e^{\mathit{polyGen}[1,0]}\} \qquad [10]$$

En el patrón estructural [9] hemos dejado fija la parte polinómica del patrón [6]. En tal situación, siempre que se generen instancias de la misma, dicho factor polinómico permanece inalterado y el único que varía es el correspondiente al exponente de la parte exponencial. Evidentemente hay otras muchas configuraciones posibles y como hemos indicado más arriba, es el profesor con sus objetivos docentes el que debe seleccionar la que estime conveniente.

Después de haber efectuado las generalizaciones correspondientes a las dos metavariables, la ventana de diseño presentará el aspecto de la ilustración 3.12. La selección de las partes de la fórmula que se desea generalizar se realiza marcando con el ratón la expresión concreta. Al pulsar sobre el botón de "pegado inteligente" se accede al diálogo de generalización en el que el profesor completará los datos relativos a la metavariable. Como se puede apreciar en la ilustración, la subexpresión generalizada en la fórmula permanece marcada. Además se agrupan los detalles de la generalización para cada metavariable declarada. Tanto la generalización en su totalidad como los detalles de la misma pueden ocultarse (como en el caso de la metavariable *u*) a conveniencia del profesor.

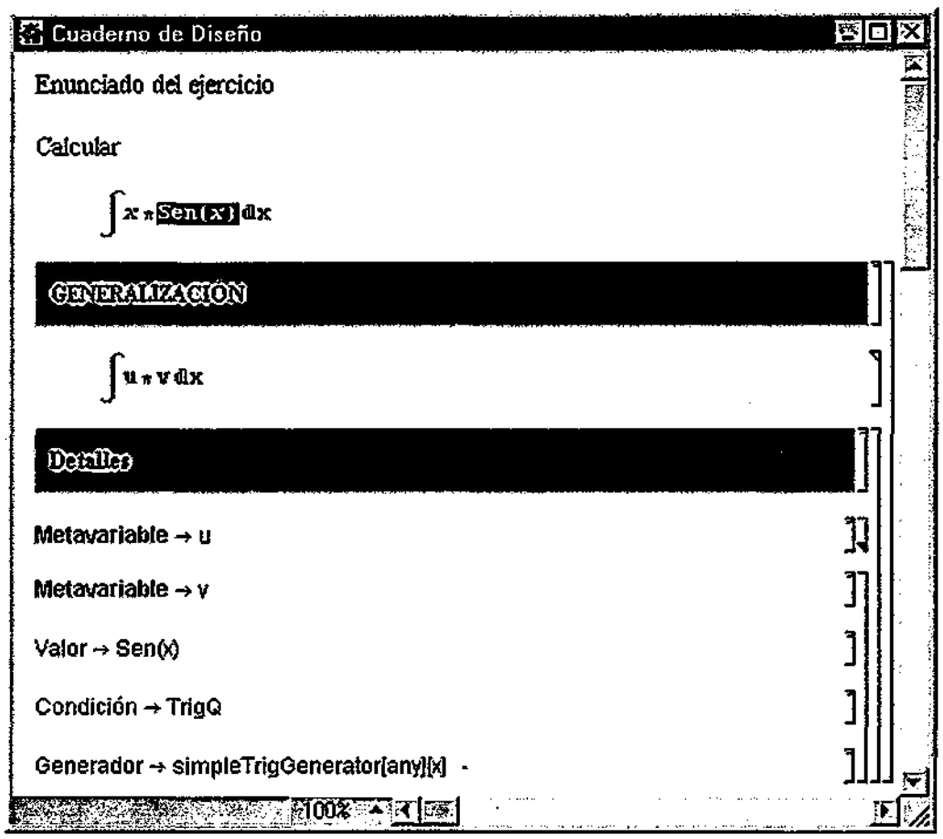

Ilustración 3.12. Cuaderno de diseño resultante tras la generalización.

Para concluir la declaración de estrategias falta introducir las acciones asociadas a las mismas. Antes de mostrar cómo se realiza dicha actividad creemos conveniente comentar algunos casos particulares en lo que se refiere a la especificación de nuevos ejercicios. Por tanto en los próximos párrafos vamos a tratar las siguientes cuestiones:

i. Cómo se declaran nuevas estrategias sobre un tipo predefinido.
ii. Cómo se declaran nuevos casos para ejercicios predefinidos.

II. Declaración de nuevas estrategias para un tipo predefinido.

Cuando el ejercicio que se va a diseñar corresponde a un tipo de ejercicio ya existente no es necesario, obviamente, volver a definir el símbolo identificador del tipo de ejercicio. La existencia de un tipo de ejercicio es condición necesaria para la existencia de alguna estrategia de resolución de algún ejercicio definido con dicho tipo. Ello implica, necesariamente, la existencia de un enunciado y de unos patrones de fórmulas.

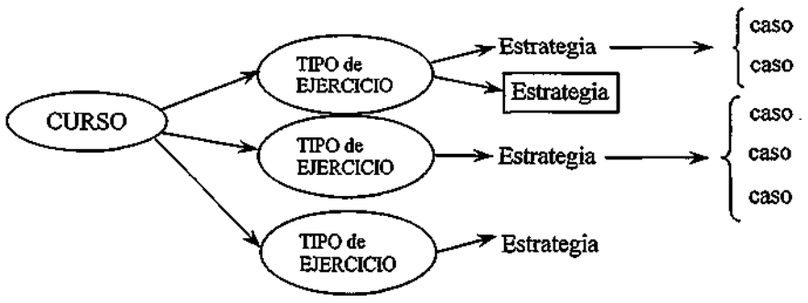

Figura 3.4. Esquema de declaración de nuevas estrategias.

Si el profesor desea introducir una nueva estrategia (figura 3.4) para un tipo de ejercicio ya existente, el sistema va a demandar del profesor que defina un nuevo patrón estructural. No olvidemos que la existencia en la estructura de un curso de un tipo de ejercicio implica la existencia de

i. un enunciado

ii. al menos una estrategia de resolución con
   - los patrones de las fórmulas definidas en el enunciado
   - una lista de acciones de resolución

En esta situación el profesor va a realizar las acciones necesarias para sustituir los elementos del punto (ii) de un ejercicio ya definido (estrategia, patrones y acciones) por los elementos correspondientes a una estrategia nueva. Sin embargo el enunciado no varía. Por este motivo el sistema presenta al diseñador un diálogo de modificación de patrón estructural (ilustración 3.13). El profesor introduce la nueva fórmula que va a utilizarse para

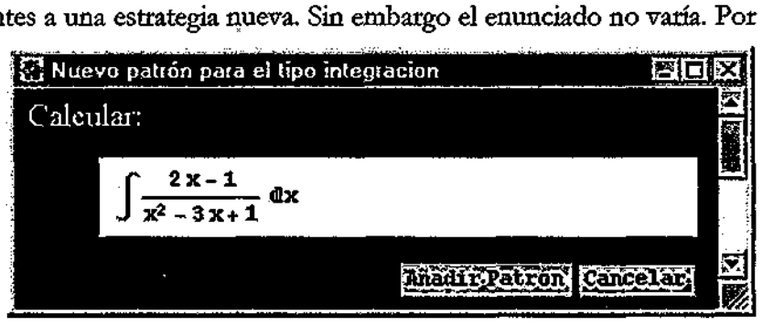

Ilustración 3.13. Nueva fórmula para integración.

representar el enunciado sin necesidad de especificar un nuevo identificador de la misma, ya que el enunciado no se modifica.

Desde el punto de vista de la estructura de datos del curso, lo que está sucediendo es lo siguiente. A partir de los datos que contemplan la estrategia de resolución *por partes* para integrales que verifiquen el patrón estructural

$$\text{Integrar}\,(u\_ \cdot v\_\,, x)$$

deseamos incorporar, para resolver ejercicios como el descrito por la fórmula de la ilustración 3.13, una estrategia cuyo modelo contemple la resolución de integrales con el patrón estructural

$$\text{Integrar}\,(\frac{P\_}{Q\_}, x)$$

Dicha estrategia se denominará, por ejemplo, *racional*. Llevará asociada una descripción de la misma, un nuevo patrón estructural de la fórmula en el enunciado, nuevas metavariables y nuevas acciones de resolución; pero siempre bajo la jerarquía del tipo de ejercicio *integración* y con un mismo enunciado común para ambos ejercicios. Por tanto el enunciado del tipo de ejercicio se hereda en la declaración de nuevas estrategias, incluyendo dicha herencia los símbolos identificadores de las fórmulas en el enunciado original.

En el apéndice A de la memoria (apartado C) se muestra la estructura de datos resultante. En tales condiciones, en tiempo de resolución, son perfectamente compatibles enunciados como pueden ser

Ejercicio 1. Calcular

$$\int 3x^2 \cdot Cos(x)dx$$

o bien

Ejercicio 2. Calcular

$$\int \frac{3x+1}{(x-3)^2}dx$$

En el caso del ejercicio 1, la ***integral*** se ha reemplazado por el patrón correspondiente a la estrategia *por partes*, realizándose las asignaciones

$$u \rightarrow 3x^2$$

$$v \rightarrow Cos(x)$$

mientras que en el segundo ejercicio, la ***integral*** se ha reemplazado por el patrón correspondiente a la estrategia *racional*, realizándose las asignaciones

$$P \rightarrow 3x+1$$
$$Q \rightarrow (x-3)^2$$

III. Definición de nuevos casos de ejercicios predefinidos

Vamos a tratar a continuación la situación en la que el profesor desea introducir un nuevo caso relativo a un ejercicio ya definido previamente (ver figura 3.5). El significado de un nuevo caso corresponde a la situación en la cual ya se ha definido una estrategia y se desea modificar la declaración de las metavariables. Por tanto las fórmulas son las mismas, la estrategia de resolución es la misma y las metavariables son las mismas salvo que se modifican bien los condicionantes, bien las funciones generadoras o ambos simultáneamente.

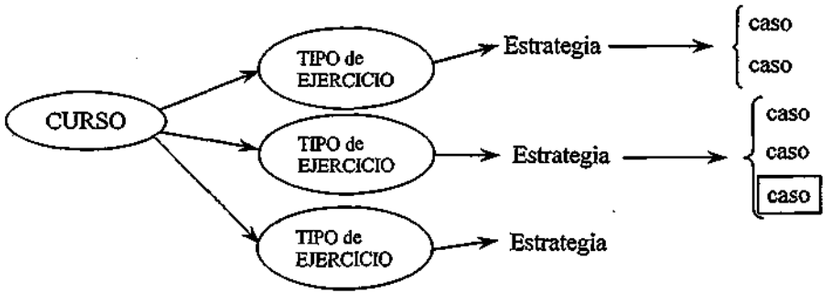

Figura 3.5. Esquema de declaración de nuevo caso.

Para analizar esta situación *MathEdu Designer* nos proporciona una interfaz en la que nos presenta los distintos tipos de ejercicios que ya están predefinidos dentro del curso. Supongamos que el profesor opta por algún tipo de los que se le enumeran. En esta situación no es necesario que defina ni el tipo de ejercicio, ni el enunciado ni la estrategia. Dichas tareas ya han sido realizadas con anterioridad y, por tanto, se obvian en esta etapa.

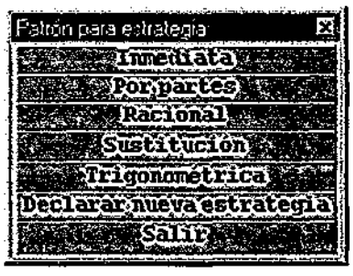

Ilustración 3.14. Opción nuevo caso

La declaración de nuevos casos para una estrategia utiliza los patrones estructurales de las fórmulas predefinidas asociadas a las estrategias. Así, antes de proceder a la nueva declaración de metavariables, se

muestra una paleta con las estrategias ya definidas (ilustración 3.14). Cualquier selección que realice el profesor (excepto la de nueva estrategia, según se vio en el punto II) despliega de forma automática el diálogo de generalización para actualizar los datos predefinidos así como una ventana similar al cuaderno de diseño con el patrón estructural que se va a redefinir (ilustración 3.15)

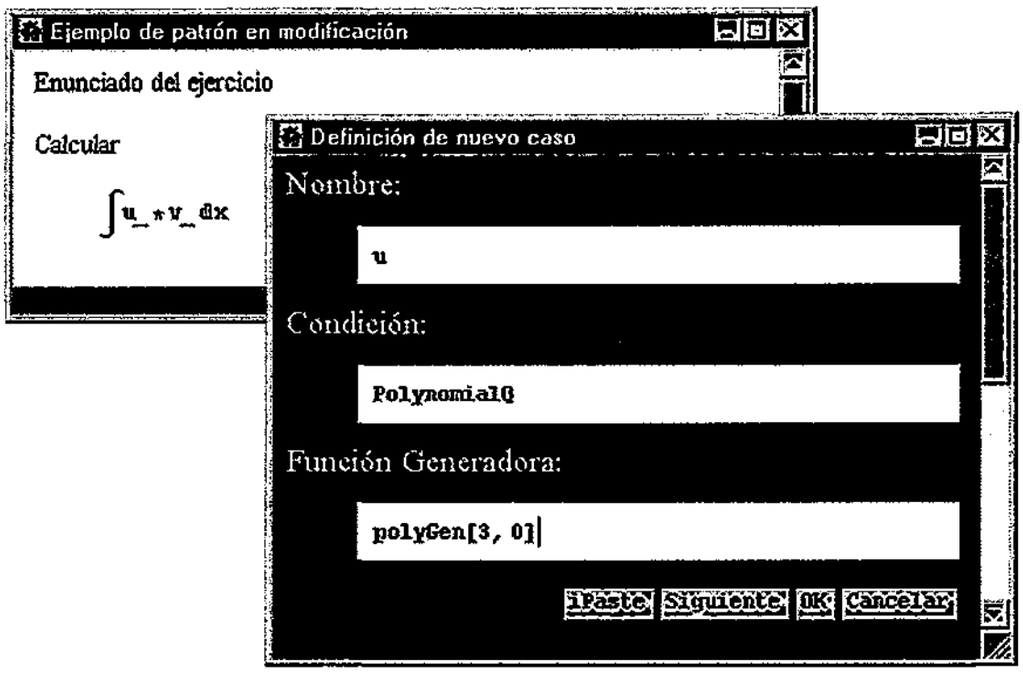

Ilustración 3.15. Redeclaración de metavariables para un nuevo caso.

Mediante diálogos similares al mostrado en la ilustración 3.15, la expresión simbólica que se muestra en el cuaderno de diseño auxiliar se actualiza con los nuevos valores, incorporándose la nueva definición de las metavariables como un nuevo caso a las listas de casos posibles para la estrategia definida con anterioridad.

Una vez concluida la generación de un nuevo caso, la estructura de datos resultante incorpora una nueva declaración de las metavariables. En el apéndice A (apartado B) se describe la estructura de datos resultante. Con el nuevo caso el efecto que se consigue es el de generar enunciados en los que el factor exponencial sustituye al trigonométrico. A modo de ejemplos podemos considerar

Ejercicio 1. Calcular

$$\int 3x^2 \cdot Cos(x)dx$$

el cual ha sido generado con las metavariables del primer caso; o bien

Ejercicio 2. Calcular

$$\int 2x \cdot e^{2x} dx$$

el cual ha sido generado con las metavariables del segundo caso (ahora aparece el factor exponencial).

Es importante recalcar que en un nuevo caso la denominación de las metavariables permanece inalterada ya que dicha denominación es la que se utiliza en la definición de los patrones de las fórmulas y, al no haberse definido una estrategia nueva (y por tanto un patrón nuevo), debe mantenerse la denominación dada en el caso inicial a las metavariables.

La fase de generalización termina aquí. Para concluir con la declaración de estrategias sólo resta declarar las acciones de resolución asociadas a las mismas. Con las acciones se completará la aportación tecnológica necesaria para la representación de los ejercicios a través de un lenguaje para la representación de los conceptos matemáticos. Queremos poner de manifiesto la versatilidad de la estructura de datos que hemos diseñado para poder generar cursos. La sencillez conceptual del modelo de representación del conocimiento creado (matemático en este caso), junto con la potencia de *Mathematica* conforman una herramienta de diseño y generación de información altamente sofisticada.

3.3.3.2.2. Definición de las tareas del alumno.

Corresponde ahora al profesor detallar las distintas acciones que el alumno ha de llevar a cabo para resolver el ejercicio que se le plantee en la fase de resolución interactiva. En el ejemplo desarrollado en el epígrafe 3.2 hemos visto cómo se le pide al alumno que vaya introduciendo los valores de las expresiones necesarias para la resolución por partes de la integral propuesta. El efecto de las acciones que debe ir realizando el alumno se obtiene gracias a una descomposición jerárquica en tareas más simples. En este sentido, una acción de resolución es un objeto predefinido que ayuda al profesor a desarrollar los pasos de resolución de un ejercicio, los cuales deberán ser realizados por el alumno en el momento de la resolución interactiva. En los epígrafes anteriores hemos descrito cómo se modela un ejercicio efectuando la transición del lenguaje formal matemático al lenguaje de representación de *MathEdu*. En el presente epígrafe el modelo del ejercicio ya está construido y, sobre su estructura deben detallarse, con el grado de generalidad que el profesor desee, las acciones de resolución. En los párrafos siguientes vamos a describir las distintas acciones de que dispone el diseñador, cuál es su efecto durante la resolución y cuáles son sus características.

Comenzaremos situando las condiciones en las que se encuentra el diseñador una vez concluido el proceso de generalización. Por un lado dispone del cuaderno de diseño, el cual presenta un aspecto similar al descrito en la ilustración 3.12. La correspondiente opción de *Acciones* de la interfaz de *MathEdu Designer* (ilustración 3.1) despliega una paleta en la que están contenidas todas las acciones de que dispone para el diseño de la resolución del ejercicio por parte del alumno o de *MathTrainer*. El diseño de las acciones es muy similar para todas ellas salvo las excepciones de las acciones *solve* y *choice*. En cualquier caso vamos a describir el significado y funcionamiento de todas las acciones disponibles.

Antes de enumerar la lista de acciones y proceder a realizar la descripción de las mismas, analizaremos la estructura de datos del curso para comprender mejor los cambios que se van a producir en la misma una vez que se hayan definido las acciones. Los cambios que las acciones van a introducir en el curso corresponden a la cabecera de *resolución de acciones* descrita en el apartado D del apéndice A. Dicha cabecera agrupa la lista de N acciones definidas por el diseñador y es, a su vez, un elemento más de la estrategia. Las acciones se van incorporando a la estructura de datos a medida que el profesor las va definiendo. Cuando concluye la definición de una acción, automáticamente se agrega a dicha lista. El diseño de un ejercicio concluye al finalizar la especificación de acciones. En tiempo de ejecución se irán presentando las acciones al alumno secuencialmente, de forma que vaya contestando a las distintas cuestiones que se le van a ir planteando. En el caso de ser *MathTrainer* el encargado de mostrar la resolución del ejercicio la situación es similar: toma la lista de acciones y las va procesando de forma secuencial. El proceso de resolución será descrito en el epígrafe 3.4 y en el 3.5 desarrollaremos la herramienta *MathTrainer*. En el presente epígrafe nos vamos a ceñir estrictamente a cuestiones relativas al diseño y, por tanto, dejaremos a un lado cuestiones propias de la resolución propiamente dicha.

Comenzamos a continuación la descripción de los distintos tipos de acciones disponibles para efectuar el diseño de la resolución interactiva o guiada.

LISTA DE TIPOS DE ACCIONES

- *asignar*. La acción *asignar* es la única acción que denominamos *acción del sistema*. Es transparente para el alumno en el proceso de resolución. Se utiliza para efectuar asignación de valores entre variables y metavariables utilizadas durante la resolución.

- *ENTRADA DE DATOS*

  - *Entrada básica de datos*. La acción *entrada básica de datos* se utiliza para pedir al alumno que introduzca una determinada expresión matemática sin restricción alguna.

- ***entrada de datos con patrón.*** La acción de *entrada de datos con patrón* se utiliza para pedir al alumno que introduzca una determinada expresión matemática la cual deberá hacer *pattern matching* con el patrón **estructural** o **condicional** que el profesor especifica en el momento del diseño.

- ***entrada de expresiones.*** La acción de *entrada de expresiones* se utiliza para pedir al alumno que introduzca una determinada expresión matemática la cual debe coincidir (salvo simplificaciones) con la expresión que el profesor especifique en el momento del diseño.

- ***mensaje.*** La acción *mensaje* se utiliza para enviar un mensaje al alumno al cuaderno de resolución.

- ***selección de alternativa.*** La acción de *selección de alternativa* se utiliza para definir una paleta cuyos botones presentan al alumno una selección de alternativas entre las que debe seleccionar la que considere correcta para la resolución del ejercicio planteado.

- ***resolución de ejercicio.*** La acción *resolución de ejercicio* se utiliza para definir un subproblema que es necesario resolver como parte del proceso principal de resolución del ejercicio planteado al alumno.

Una vez enumeradas los tipos de acciones vamos a describir los diálogos que utiliza el profesor para cada una de las acciones de la lista anterior. En algunos casos el diálogo es muy simple e intuitivo. En otros (*selección de alternativa* y *resolución de ejercicio*), para efectuar el diseño es necesario encadenar varios diálogos y el proceso es considerablemente más complejo. Por ese motivo dejaremos estos dos casos para el final con el fin de habernos familiarizado algo más con la terminología utilizada. Utilizaremos en todos los casos una representación gráfica que facilitará su descripción.

Acción de asignar

Como ya indicamos anteriormente la acción de asignación es una acción transparente para el alumno. Sirve para asignar valores entre variables. Podemos representar gráficamente la acción mediante la tabla 3.3 siguiente. El campo en color indica la acción y el resto de campos los argumentos sobre los que se aplica.

Asignar
variable
valor

Tabla 3.3. Acción *asignar*.

donde

- *variable* representa el identificador de la variable a la cual se va a asignar el valor,

- *valor* contiene el valor a asignar, que puede ser cualquier expresión, pudiendo depender inclusive de algunas metavariables. Obviamente *valor* puede contener, a su vez, el identificador de otras variables que determinen su valor durante la resolución del ejercicio.

No debe pasarnos por alto el hecho de que el *valor* que se asigna a la *variable* está protegido para prevenir que pueda ser evaluada dicha variable y, por consiguiente, se pierda el valor que se desea copiar. Pensemos por ejemplo que el valor a asignar a una variable es

$$\int x^2 dx$$

al asignar el valor, si no lo protegemos convenientemente, el valor que tomaría la *variable* sería

$$\frac{x^3}{3}$$

pues, como ya hemos explicado anteriormente, *Mathematica* evalúa siempre que puede hacerlo todas las expresiones que acceden al núcleo. Por este motivo es necesario que el contenido de la variable esté protegido ante cualquier evaluación no deseada por el diseñador. Obviamente el resultado de una evaluación no planificada por el diseñador podría tener un efecto impredecible en el tratamiento que se diese a la información que supuestamente debe contener la *variable*.

El diálogo correspondiente al diseño de este tipo de acción se muestra en la ilustración 3.16. Como ejemplo hemos colocado, como *variable* el identificador *solución* y, como valor que debe tomar dicha variable, el cuadrado del valor absoluto de la variable *resultado*.

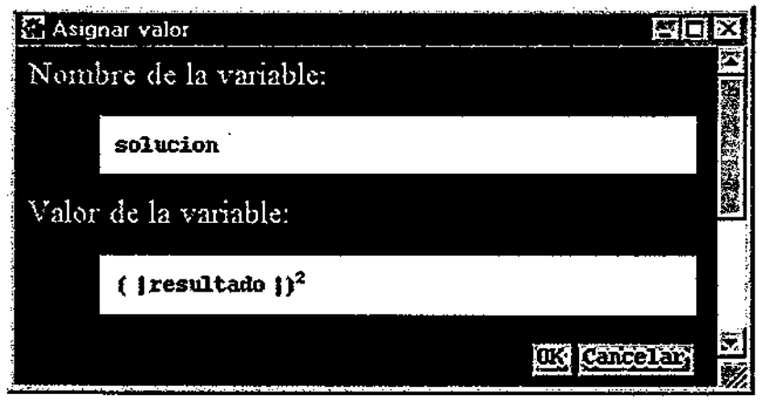

Ilustración 3.16. Acción asignar.

Durante la resolución la variable *resultado* tomará un valor y, en el momento en que el motor de inferencia de *MathEdu Solver* o de *MathTrainer* deba ejecutar la acción *asignar*, se evaluará la operación $|resultado|^2$ y el valor resultante de dicha operación se asignará sobre la variable *solución*.

ACCIONES DE ENTRADA DE DATOS

Las *acciones de entrada de datos* sirven para que el alumno introduzca en el sistema expresiones matemáticas. Como hemos indicado al inicio de este apartado puede haber tres tipos de entrada de acciones. Estos son:

Acción de entrada básica de datos

La acción de *entrada básica de datos* sirve para que el alumno introduzca en el sistema una expresión matemática cualquiera. Es la más simple de las acciones de introducción de datos por parte del alumno, ya que no se hace ningún tipo de comprobación del dato tecleado. La tabla 3.4 siguiente esquematiza la acción y sus argumentos

entrada básica de datos
descripción
variable

Tabla 3.4. Acción *entrada de datos*.

donde

- *descripción* contiene una descripción textual de la operación que debe realizar el alumno,
- *variable* representa el identificador de la variable a la cual se va a asignar el valor que teclee el alumno.

El diálogo correspondiente a esta acción se muestra en la ilustración 3.17. En el ejemplo hemos considerado que la expresión que teclea el alumno se almacenará en la variable *u*. Durante la resolución la variable *u* almacenará el valor que teclee el alumno, sin efectuar ninguna comprobación sobre el mismo.

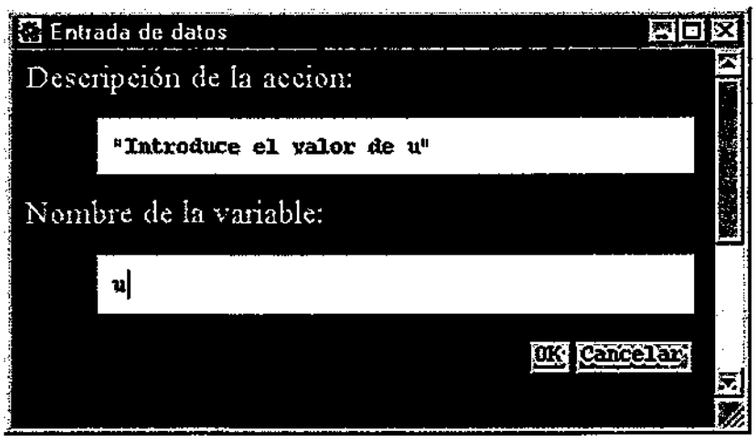

Ilustración 3.17. Acción de entrada básica de datos.

Acción de entrada de datos con patrón

La acción de *entrada de datos con patrón* sirve para que el alumno introduzca en el sistema una expresión matemática la cual debe verificar una condición de patrón estructural o condicional impuesta por el profesor. La tabla 3.5 siguiente esquematiza la acción y sus argumentos

Entrada de datos con patrón
descripción
variable
condición

Tabla 3.5. Acción *entrada de datos con patrón.*

donde

- *descripción* contiene una descripción textual de la operación que debe realizar el alumno,
- *variable* representa el identificador de la variable a la cual se va a asignar el valor que teclee el alumno,
- *condición* contiene una condición impuesta por el profesor que debe cumplir la expresión que teclee el alumno. El contenido del campo *condición* puede representar cualquier abstracción matemática.

El diálogo correspondiente al diseño de esta acción se muestra en la ilustración 3.18. En consonancia con el ejemplo descrito en el epígrafe 3.2.2 mostramos la cuarta acción asociada a la resolución que allí se muestra. Se declara como *variable* el símbolo identificador *V*. Como *condición* de la expresión introducida por el alumno se exige que dicha expresión debe ser la integral de la función *v* que el alumno ya habrá introducido previamente.

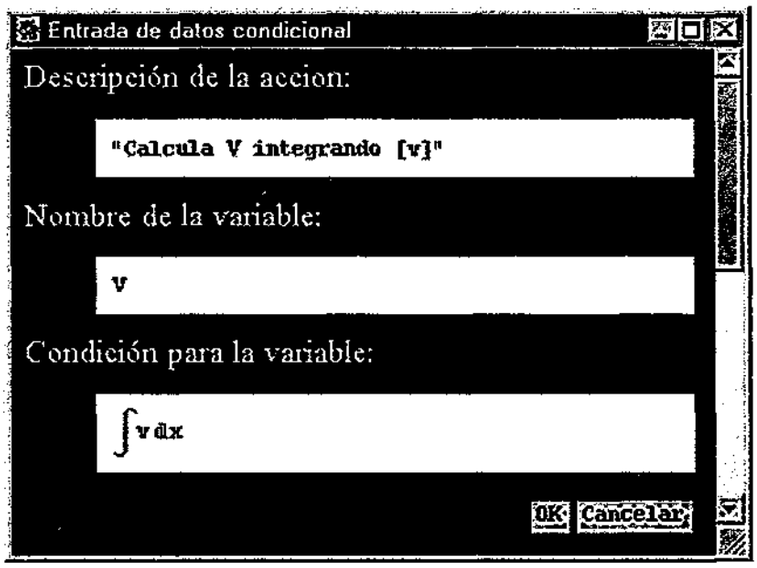

Ilustración 3.18. Acción de entrada de datos condicional.

Durante la resolución la variable *V* tomará el valor que teclee el alumno siempre y cuando dicha expresión verifique la condición impuesta por el profesor. Caso contrario se le avisará al alumno de que la expresión tecleada no es correcta.

Acción de entrada de expresiones

La acción de *entrada de expresiones* sirve para que el alumno introduzca en el sistema una expresión matemática la cual debe ser equivalente a la expresión declarada por el profesor. La tabla 3.6 siguiente esquematiza la acción y sus argumentos

entrada de expresiones
descripción
variable
expresión

Tabla 3.6. Acción *entrada de expresiones*.

donde

- *descripción* contiene una descripción textual de la operación que debe realizar el alumno,

- *variable* representa el identificador de la variable a la cual se va a asignar el valor que teclee el alumno,

- *expresión* contiene una expresión simbólica formulada por el profesor. La expresión que teclee el alumno debe ser semejante a esta expresión, salvo posibles simplificaciones. Las simplificaciones se realizan en *Mathematica* mediante las primitivas **Simplify** y **FullSimplify**.

El diálogo correspondiente al diseño de esta acción se muestra en la ilustración 3.19. Se muestra el diseño de la tercera acción asociada a la resolución que allí se muestra. Se declara como *variable* el símbolo identificador *du*. El dato que teclee el alumno debe ser compatible con la *expresión* esperada (en este caso la derivada de *u*, variable asignada por el alumno en la acción de entrada básica de datos anterior).

Durante la resolución la variable *u* tomará el valor que el alumno teclee (pregunta 1). La expresión que teclee el alumno como respuesta a esta nueva pregunta debe coincidir (salvo simplificaciones) con la derivada de dicha variable. El valor tecleado por el alumno se almacena en la variable *du*. En caso de que el valor tecleado no sea correcto se avisará al alumno de la contingencia.

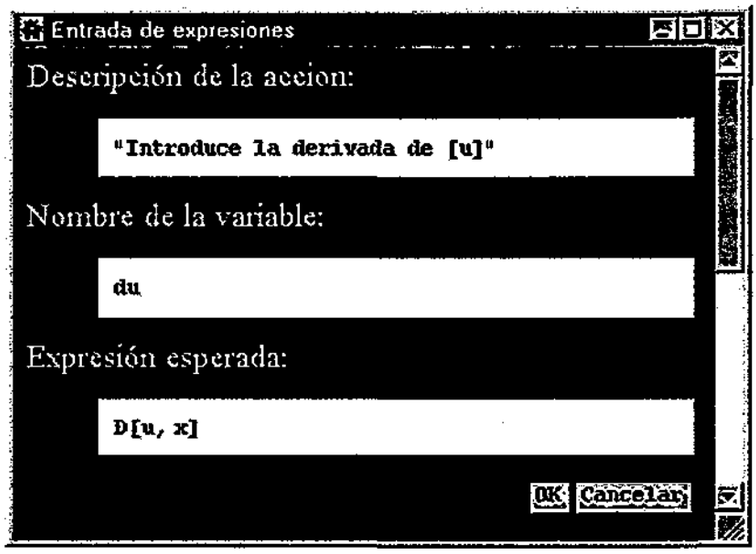

Ilustración 3.19. Acción de entrada de expresiones.

Las dos últimas acciones que acabamos de exponer sobre entrada de datos realizan una comprobación de los datos que introduce el alumno en el sistema y le advierten en caso de error, permitiéndole volver a introducir nuevamente la expresión. En caso de error reiterado el sistema puede tomar el control de la resolución invocando al sistema *MathTrainer* para que sea este el que muestre al alumno cómo debe realizar el ejercicio.

Acción de mostrar mensaje

La acción de *mostrar mensaje* sirve para comunicar al alumno un mensaje de texto cualquiera. La tabla 3.7 siguiente esquematiza la acción y sus argumentos

mostrar mensaje
Mensaje

Tabla 3.7. Acción *mostrar mensaje*.

donde

- *mensaje* contiene el mensaje para el alumno.

El diálogo correspondiente a esta acción se muestra en la ilustración 3.20. En el ejemplo se muestra el mensaje para el alumno que aparece en la ilustración 3.6, previo a la selección de una alternativa en la paleta de selección.

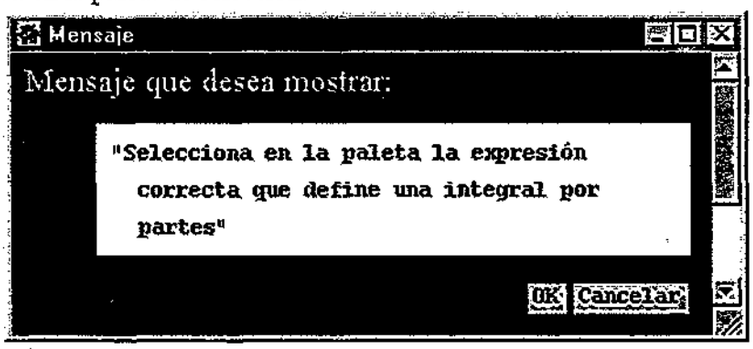

Ilustración 3.20. Acción mensaje.

Acción de selección de alternativa

La acción *selección de alternativa* sirve para mostrar al alumno una paleta de opciones a fin de que este escoja una de ellas. Por tanto, el objetivo que el diseñador persigue con esta acción es construir una paleta como la que se muestra en la ilustración 3.21, en la que el alumno dispone de diferentes alternativas de selección ligadas a los distintos botones de la misma. Cada botón representa una fórmula y se pide al alumno que escoja una de las tres para determinar si conoce o no la teoría asociada correspondiente. Cualquiera de las dos opciones incorrectas le muestran el mensaje que haya definido el profesor. La opción correcta continuará el ejercicio solicitando al alumno que introduzca la derivada de la función *u*, la cual habrá sido previamente definida mediante la correspondiente acción de *entrada de expresiones*.

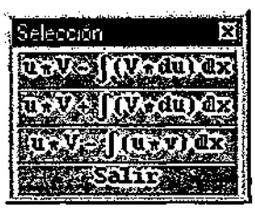

Ilustración 3.21. Opciones

La tabla 3.8 siguiente esquematiza la acción y sus argumentos

selección de alternativa
descripción$_1$, *acción*$_{11}$,..., *acción*$_{1n}$
descripción$_2$, *acción*$_{21}$,..., *acción*$_{2m}$
...
descripción$_p$, *acción*$_{p1}$,..., *acción*$_{pq}$

Tabla 3.8. Acción *selección de alternativa.*

donde

- *descripción* contiene una descripción textual de la operación que debe realizar el alumno,
- *acción*$_{ij}$ corresponde al código de cualquiera de las acciones descritas en el presente epígrafe.

El diálogo correspondiente al diseño de esta acción se muestra en la ilustración 3.22. Como ejemplo hemos introducido como *descripción* la expresión errónea de la fórmula para resolver una integral por partes. El objetivo es construir una paleta en la que aparezcan, por ejemplo, tres fórmulas similares (ilustración 3.21). Dos de ellas son erróneas y la acción asociada es un mensaje de error para el alumno. La opción correcta (descrita en esta ilustración) lleva asociada una acción de *entrada de expresiones* y otra de asignación de variables.

Se puede observar en la ilustración el botón etiquetado *Acciones.* Este botón sirve para desplegar la paleta de acciones y que así el profesor, en el momento del diseño, tenga fácilmente accesible la lista de todas las acciones disponibles para que seleccione las que estime oportuno.

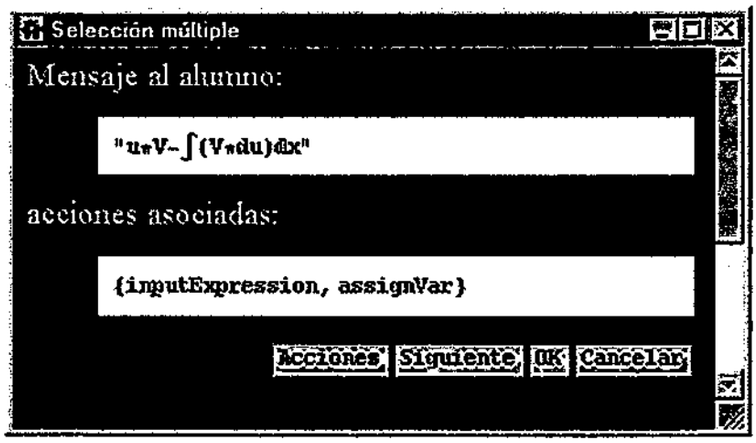

Ilustración 3.22. Acción de selección de alternativa.

Hemos de indicar que el proceso de definición de una acción de tipo *selección de alternativa* es iterativo. Como hemos podido comprobar en la ilustración 3.22, hay tres botones en el diálogo de la acción. Los relevantes son

- ***Siguiente.*** Efectúa las dos tareas siguientes:

i. Limpia el contenido del diálogo para poder definir la siguiente opción de la acción de *selección de alternativa,*

ii. Muestra los diálogos correspondientes a las acciones que se van seleccionando en la paleta de tipos de acciones para proceder a efectuar su correspondiente definición. Se van anotando en el campo *acciones asociadas.*

- ***OK*** Efectúa las dos tareas siguientes:

i. Muestra los diálogos correspondientes a las acciones que se van seleccionando en la paleta de tipos de acciones para proceder a efectuar su correspondiente definición. Se van anotando en el campo *acciones asociadas,*

ii. Cierra el diálogo de *selección de alternativa.*

Para construir la paleta mostrada anteriormente el diseñador debe usar el diálogo de la ilustración 3.22 tres veces. En las dos primeras pulsará el botón ***Siguiente*** para avanzar en la definición de las opciones. Finalmente, cuando haya definido la acción correspondiente a la última opción saldrá pulsando ***OK***

El código resultante en la estructura de datos de *MathEdu,* que sigue el esquema indicado en la tabla 3.8, se muestra en el apartado E del apéndice A.

Acción de resolución de ejercicio

La acción *resolver ejercicio* es, con seguridad, la más compleja de todas las descritas hasta el momento. Es compleja tanto en su definición, que involucra tres pasos con otros tantos diálogos asociados, como en el momento de la ejecución. Ahora vamos a centrarnos exclusivamente en la declaración de la acción.

La acción *resolver ejercicio* sirve para iniciar la resolución de un subproblema del ejercicio que está siendo resuelto. Si pensamos en la resolución de un ejercicio como en la selección de una rama del árbol de posibles ramas de resolución, esta acción va a permitir al profesor crear otras ramas que nacen de la principal en dicho árbol. Ya vimos en la figura 3.2 esta circunstancia y cómo se utilizó para resolver el subproblema asociado al ejercicio que se estaba resolviendo.

La tabla 3.9 siguiente esquematiza la acción y sus argumentos

resolver ejercicio
descripción
tipo
entrada(...)
salida(...)

Tabla 3.9. Acción *resolución de ejercicio.*

donde

- *descripción* contiene una descripción textual de la operación que debe realizar el alumno,
- *tipo* corresponde al tipo del problema que se debe resolver como parte de la resolución del ejercicio principal,
- *entrada* contiene los datos de entrada para el subproblema,
- *salida* contiene los datos de salida del subproblema que se pasan al ejercicio principal.

Para definir una acción de *resolver ejercicio* se requiere efectuar secuencialmente los siguientes tres pasos

1. El de definición del *tipo* de ejercicio a resolver,
2. El de definición de los datos de *entrada* del nuevo ejercicio,
3. El de definición de los datos de *salida* del nuevo ejercicio.

Vamos a describir a continuación cada uno de los pasos indicados. Para facilitar en alguna medida la explicación vamos a considerar que el profesor está diseñando la resolución del siguiente ejercicio de aritmética de fracciones

Ejercicio. Calcular $\left(\frac{2}{3}+\frac{1}{5}\right)\cdot\frac{4}{3}$

Para resolverlo el profesor desea guiar al alumno sugiriéndole que resuelva la expresión entre paréntesis efectuando la suma correspondiente y que posteriormente calcule el resultado de la suma anterior multiplicado por el factor fraccionario.

El ejercicio consta de las metavariables

$\{n1, RationalQ, rationalGen\}$ que representa el primer sumando, es decir $\frac{2}{3}$,

{*n2, RationalQ, rationalGen*} que representa el segundo sumando, es decir $\frac{1}{5}$,

{*factor2, RationalQ, rationalGen*} que representa el segundo factor, es decir $\frac{4}{3}$.

Mediante la acción *resolver ejercicio* el profesor va a definir una nueva variable del subproblema, a la que denominará *resultadoSuma*, y que contendrá el resultado de efectuar la operación que se indica en el mismo. Tal operación corresponde en este caso a una suma. Es, por tanto, un ejercicio de tipo *adición*. El profesor declarará una acción *resolver ejercicio* para relacionar el subproblema con dicho tipo. Las metavariables en los ejercicios de tipo *adición* se referencian mediante los identificadores *sumando1* y *sumando2*. En el ejercicio nuevo las metavariables *n1* y *n2* juegan el papel de estas. Además pueden hacerlo indistintamente (por la conmutatividad de la suma).

Por último, en el tercer paso se realiza la identificación entre variables que contienen los resultados de ambos problemas. Supondremos que el subproblema almacena el resultado de la suma en la variable *resultadoSuma* y su contenido se pasará a la variable *factor1* del ejercicio principal. La variable *factor1* es la que ha escogido el profesor para representar la suma que aparece en el problema principal. En el apéndice A (apartado F) se muestra en un esquema al nivel de la estructura de datos el tipo de relaciones existentes entre ambos problemas. Ahora vamos a ir describiendo con detalle la interfaz que permite al profesor realizar cada uno de los pasos a que acabamos de referirnos.

PASO 1. Definición del tipo de ejercicio.

Para indicar el tipo del ejercicio que el alumno debe resolver como parte de la resolución del ejercicio principal, el profesor dispone del diálogo *Resolver subproblema* mostrado en la ilustración 3.23. El profesor dispone de un campo para especificar un mensaje que se mostrará al alumno en el momento de iniciar la resolución del subproblema y otro campo para especificar el *tipo de ejercicio* correspondiente al subproblema que desea definir.

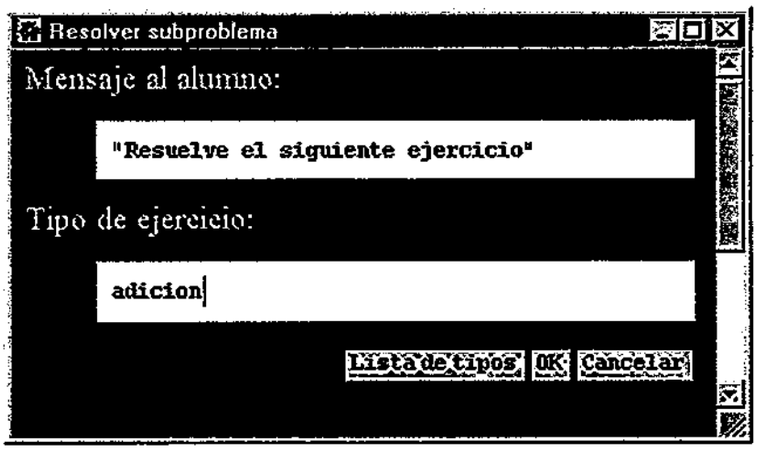

Ilustración 3.23. Declaración del tipo de ejercicio en la acción *resolver ejercicio*.

El profesor dispone de la posibilidad de listar los tipos de ejercicios prediseñados en el curso mediante la opción de *Lista de tipos*. El diseño de un subproblema siempre debe hacerse en función de un ejercicio existente en el curso y al cual puede hacerse referencia como parte de la resolución de un ejercicio más complejo.

Al concluir esta fase se genera el siguiente seudocódigo parcial para la estructura de datos de *MathEdu*

resolverEjercicio("Resuelve el siguiente ejercicio", adición)

y se accede al siguiente paso.

PASO 2. Definición de los datos de *entrada* del nuevo ejercicio.

En este paso el profesor debe introducir los datos de entrada del subproblema. Más concretamente debe establecer la relación existente entre los datos (metavariables) que está utilizando en el diseño del ejercicio principal y la fórmula con que hubo definido el ejercicio que ahora va a actuar como subproblema. Para realizar esta identificación el profesor dispone del diálogo *Datos de entrada de subproblema* descrito en la ilustración 3.24

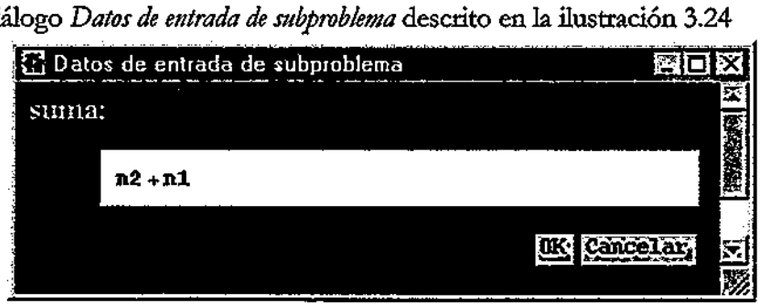

Ilustración 3.24. Definición de los datos de *entrada*.

El diseño de este diálogo es, pese a su aparente sencillez, de cierta dificultad conceptual. Si observamos el diálogo, consta de tres grupos básicos de elementos: una cabecera, un campo de edición de texto y un grupo de botones. Los dos primeros aparecen tantas veces como fórmulas haya definidas en el subproblema. Las cabeceras especifican los nombres de los identificadores de las fórmulas en el ejercicio previamente definido. En el caso del ejemplo sólo existe una fórmula cuyo símbolo identificador es *suma*. Los campos de edición de texto permiten especificar una fórmula (o fórmulas, en el caso de que existan varias). El profesor debe introducir en dicho campo una expresión que se utiliza para construir el enunciado del nuevo problema a resolver.

Al concluir esta fase se accede al paso final y se actualiza el código para la estructura de datos de *MathEdu* añadiendo la parte correspondiente a este segundo paso (el cual resaltamos en negrita).

resolverEjercicio("Resuelve el siguiente ejercicio",
adición,
entrada(suma → n2 + n1))

PASO 3. Definición de los datos de *salida* del nuevo ejercicio.

En este paso el profesor debe introducir los datos de salida del subproblema hacia el ejercicio inicial. En concreto, debe establecer la relación existente entre los resultados (variables) del subproblema y las variables que los representan en el ejercicio inicial. Para realizar esta identificación el profesor dispone del diálogo *Datos de salida de subproblema* descrito en la ilustración 3.25.

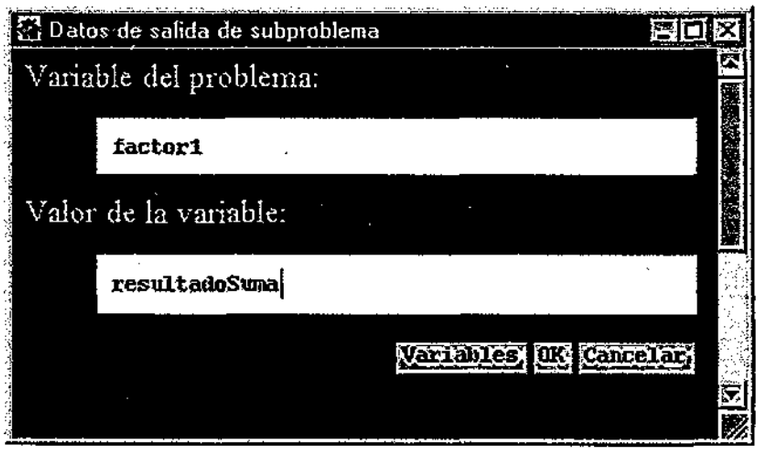

Ilustración 3.25. Definición de los datos de *salida*.

En este diálogo el profesor debe realizar la identificación de variables que contienen resultados. En primer lugar especifica, en el campo correspondiente a la *variable del problema*,

la variable que representa el resultado en el ejercicio inicial. En este caso *factor1* hace referencia al contenido de dicha parte en el ejercicio que se pretende resolver.

Por su parte, el segundo campo corresponde a una expresión que incluye a la variable del subproblema que contiene el resultado parcial correspondiente al ejercicio principal y cuyo valor debe comunicarse a este para poder continuar con la resolución. Podemos observar que existe un botón, etiquetado *Variables*, el cual nos permite acceder a la lista de identificadores de las variables del subproblema.

Al concluir esta fase se concluye el diseño de la acción *resolver ejercicio*. Nuevamente se actualiza el código para la estructura de datos de *MathEdu* añadiendo la parte correspondiente a este último paso (el cual resaltamos en negrita).

resolverEjercicio("Resuelve el siguiente ejercicio",
adición,
entrada[suma → n2 + n1],
salida(factor1 → resultadoSuma))

Como hemos podido comprobar, la dificultad conceptual y de diseño de la acción de *resolución de ejercicio* es elevada. Por el contrario, la representación simbólica de los datos necesarios para referirse a un ejercicio previamente definido desde un ejercicio que actualmente se está definiendo, es de una gran sencillez. Se combina, por tanto, de un lado la claridad en la concepción de un método de resolución y de otro la complejidad del diseño para dicho método.

Hasta aquí hemos venido exponiendo los distintos tipos de acciones de resolución. La declaración de las distintas acciones dentro del contexto de un ejercicio y su generalización tienen su reflejo en el cuaderno de diseño. Ya vimos en la ilustración 3.12 el aspecto que tenía dicho cuaderno tras la generalización del ejercicio mediante la declaración de las oportunas metavariables. En la ilustración 3.26 se muestra el cuaderno de diseño durante la declaración de las acciones de resolución del ejercicio. El aspecto del cuaderno de diseño es muy similar al que tiene el cuaderno de resolución (ver ilustración 3.6) y en su elaboración toma auténtica relevancia la programación por demostración, de la que ya hemos hablado en epígrafes anteriores. El profesor mediante operaciones de "pegado inteligente" transfiere datos entre las fórmulas y variables del ejercicio, siendo éste uno de los paradigmas propios de dicho tipo de programación.

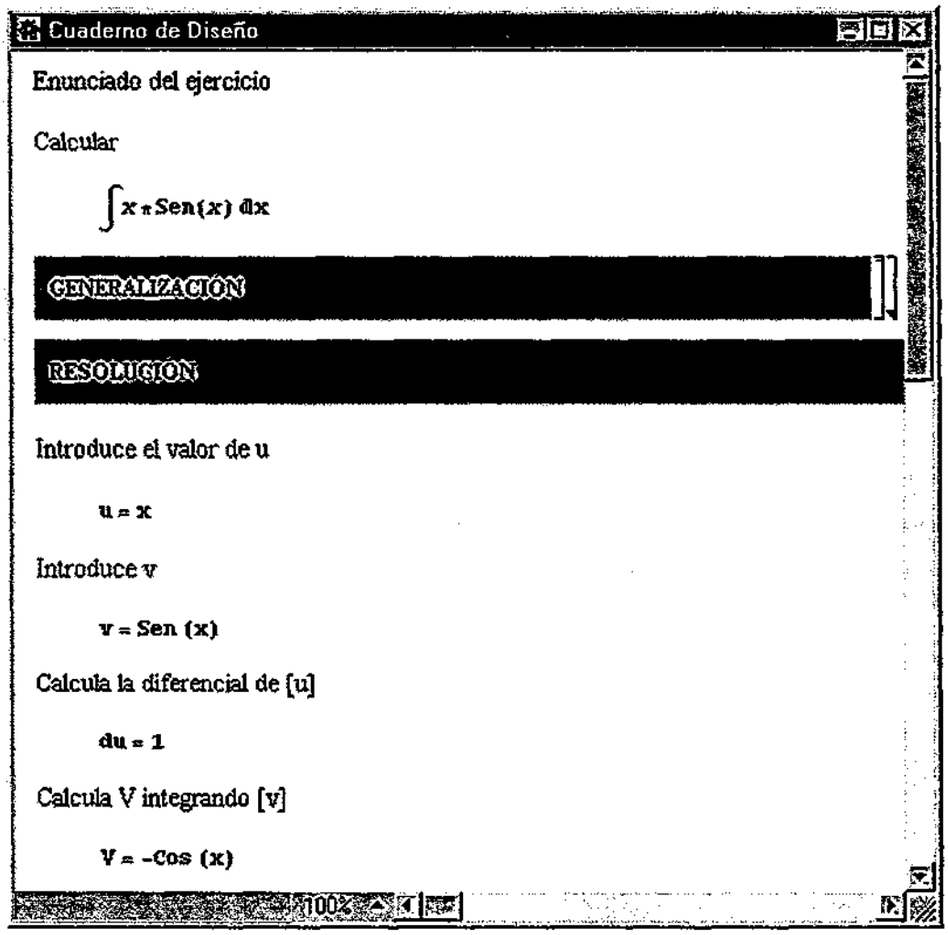

Ilustración 3.26. Cuaderno de Diseño durante la declaración de acciones de resolución.

Además de las acciones de resolución propiamente dichas que acabamos de detallar existen otras dos acciones transparentes para el diseñador que se añaden de forma automática a la lista de acciones a ejecutar en la resolución de un ejercicio. Estas acciones son:

Acción de fin de subproblema

La acción de *fin de subproblema* sirve para indicar al motor que el subproblema que estaba resolviéndose ha concluido. Libera las variables y metavariables utilizadas, cambia de contexto de resolución y continúa con las acciones pendientes de ejecución. Se agrega de forma automática, transparente para los usuarios, a la lista de acciones que introduce el profesor, cuando este concluye con el diseño de una acción de *resolución de ejercicio.*

El código que genera la acción para la estructura de datos del curso es

finSubproblema( )

Acción de fin de resolución

La acción de *fin de resolución* sirve para indicar al motor que el ejercicio que estaba resolviéndose ha concluido. Libera las variables y metavariables utilizadas y avisa mediante un mensaje al alumno que el ejercicio ha concluido. Esta acción se añade de forma automática, transparente para los usuarios, en tiempo de ejecución, cuando se leen las acciones diseñadas del tipo de ejercicio que se le presenta al alumno para que sea resuelto y antes de ejecutar el bucle que va presentándolas secuencialmente.

El código que genera la acción para la estructura de datos del curso es

finResolucion( )

3.3.3.3. Conclusión

Termina aquí la parte correspondiente al diseño de ejercicios con la herramienta de autor *MathEdu Designer*. A lo largo de las páginas precedentes hemos ido desgranando los distintos pasos necesarios para modelar el conocimiento contenido en un ejercicio de cálculo (conocimiento tanto intrínseco como extrínseco), en una estructura de datos que, a nuestro juicio, resulta de una llamativa sencillez, dado el tipo de conceptos complejos que se manejan.

Resulta evidente que la interfaz que nos proporciona *Mathematica* no es en absoluto cómoda, desde el punto de vista de los objetos que se manipulan. Ahora bien, la facilidad de edición gráfica de símbolos matemáticos y, sobre todo, la posibilidad de efectuar programación simbólica así como *pattern-matching* de las expresiones simbólicas que manipulamos, son razones más que suficientes para decantarse por una herramienta de estas características al menos para la elaboración de un prototipo. En cualquier caso somos conscientes de que el intento de distribución de una herramienta de estas características para su uso fuera del ámbito de la investigación, en situaciones reales de didáctica y educación Matemática, requiere de un proceso de migración hacia aplicaciones más versátiles, de un uso más cómodo e intuitivo, con abundantes ayudas, etc. En el trabajo que resume la presente memoria nos hemos ceñido al aspecto puramente conceptual de la investigación sobre métodos de representación del conocimiento matemático y su utilización en herramientas de cálculo simbólico. Obviamente para facilitar el diseño de las representaciones de los ejercicios ha sido necesario construir una interfaz que posibilitase esto, pero aún consideramos que estamos a medio camino del objetivo de conseguir que la herramienta sea utilizada por cualquier profesor en cualquier situación docente.

Por último queremos hacer una referencia al propio proceso de diseño en sí mismo. Hemos tratado de realizar, con las limitaciones de la interfaz ya comentadas, que la herramienta de autor sea de un uso sencillo para el profesor que debe usarla. Somos conscientes de que en ocasiones la dificultad de diseño es elevada, entre otras consideraciones por el hecho de estar basada en la programación simbólica con *Mathematica*. Es conveniente que el profesor esté familiarizado con dicho sistema de cálculo aunque no es imprescindible. Sí que es tal vez más importante poseer una alta capacidad de abstracción para ser capaces de planificar la resolución de un ejercicio, sus elementos relevantes (fórmulas, metavariables, acciones de resolución), la estrategia (o estrategias) válidas para la resolución, etc. Es fundamental realizar, previamente al diseño de un ejercicio, una resolución del mismo con lápiz y papel, tomando nota de todos los detalles que resulten relevantes y que, posteriormente, resultarán de utilidad para el diseño. Evidentemente, con la práctica y el uso continuado de la herramienta de autor, puede llegar a realizarse el diseño de ejercicios directamente, sin realizar el paso previo de la resolución en papel.

3.4. Módulo de resolución interactiva: *MathEdu Solver*

Una vez que hemos concluido con la fase de diseño de ejercicios abordamos la parte correspondiente a la resolución interactiva de los mismos con el alumno. El motor de resolución de *MathEdu Solver* está compuesto de tres partes:

1. Un generador aleatorio de ejercicios

2. Un módulo decisor de las estrategias de resolución válidas para un ejercicio

3. Un módulo gestor de las acciones del alumno

Recordemos que el cuaderno de resolución se utiliza para mostrar al alumno el ejercicio que se le propone y los distintos pasos y acciones que va efectuando. En él se le presenta el enunciado y se le van dando distintas indicaciones como pueden ser :

- que debe escoger una opción de entre las que se le ofrecen en una paleta de estrategias o de acciones,
- se le envían mensajes para introducir un dato o una fórmula,
- se le indica si el dato introducido es o no correcto,
- se le presentan los resultados parciales que va obteniendo, etc.

En el epígrafe 3.2.2 se ejemplificó la resolución de un ejercicio utilizando *MathEdu Solver*. En la ilustración 3.6 se mostró el aspecto general que muestra el cuaderno de resolución durante el proceso, mostrando el enunciado, mensajes del sistema y respuestas del alumno. Además, puesto que los ejercicios propuestos pueden apoyarse a su vez en otros ejercicios, se utiliza un cuaderno de resolución por cada ejercicio que deba resolverse. Remitimos al lector al ejemplo indicado para recordar estos aspectos en los que en este epígrafe vamos a profundizar.

En los siguientes apartados vamos a detallar las tres partes a que hemos hecho referencia, tratando en cada momento de mantener la explicación sin perder de vista la equiparación conceptual entre los procesos seguidos por la herramienta y los seguidos por el alumno.

3.4.1. Generador aleatorio de ejercicios

El generador aleatorio de ejercicios se activa cuando el alumno selecciona el sistema de resolución *Solver* de *MathEdu*. Se invoca en ese instante la función de selección aleatoria de

ejercicios, la cual, de forma muy simple efectúa múltiples acciones para obtener el ejercicio que se ha de presentar al alumno en el cuaderno de resolución.

El proceso consiste en seleccionar, de forma aleatoria, el tipo de un ejercicio de entre todos los tipos de ejercicios que se están practicando en ese momento. Una vez que se ha seleccionado un tipo de ejercicio el enunciado ya queda unívocamente especificado (recordemos que para cada tipo de ejercicio existe en la estructura de datos un único enunciado). No sucede así con las estrategias, ya que para el enunciado correspondiente al tipo escogido, pueden existir distintas estrategias de resolución, en función de los datos concretos del ejercicio. Por tanto es necesario, a su vez, seleccionar una estrategia de entre todas las posibles correspondientes al tipo escogido. Este proceso también se realiza de forma aleatoria.

Concluidas las dos operaciones básicas anteriores el sistema debe generar las metavariables que aparecen definidas en las fórmulas asociadas al enunciado. Para ello debe acceder a las metavariables correspondientes a la estrategia escogida del tipo de ejercicio seleccionado, tomar sus funciones generadoras y generar los valores correspondientes (teniendo en cuenta, claro está, la posible existencia de diferentes casos de un mismo patrón estructural como se explicó en 3.3.3.2.1-III). Dichos valores se sustituyen en los identificadores de las metavariables y se aplican las reglas correspondientes de los *patrones de fórmula* con los que se inicializan los identificadores de las mismas. Una vez concluido este proceso ya se puede enviar el enunciado al cuaderno de resolución, sustituyendo en el mismo los identificadores de las fórmulas por los valores generados.

En el caso de la ilustración que se muestra como ejemplo de cuaderno de resolución (ilustración 3.27), el enunciado corresponde a una integral cuyo integrando es un polinomio. El integrando es el resultado de sustituir el símbolo identificador de la metavariable correspondiente en la representación simbólica de la integral por el valor obtenido tras la invocación de la función generadora asociada a dicha metavariable. Dicha sustitución se realiza, como acabamos de indicar, mediante la aplicación de las reglas que relacionan metavariables y fórmulas. Podemos esquematizar esto de la siguiente forma:

1. Para obtener el valor de **fórmula_integral**

   fórmula_integral ← Integrar(*polinomio_*, *x*)

2. el sistema determina que la fórmula de la integral depende de la metavariable *polinomio*

3. seguidamente obtiene la función generadora de la metavariable *polinomio*. Esta función genera polinomios del grado que se especifique, con o sin término independiente y con coeficientes enteros aleatorios menores o iguales que 10.

polinomio ← generador_de_polinomios(
grado: natural,
termino_independiente: boolean,
indeterminada: x)

en el ejemplo *polinomio* = $1 + 5x - x^2$

4. finalmente el sistema sustituye el valor generado del *polinomio* en la fórmula de la integral

fórmula_integral ← Integrar($1 + 5x - x^2$, x)

Este proceso se aplica cada vez que el sistema debe mostrar al alumno la expresión de una fórmula correspondiente a un enunciado.

Una vez que se construye el enunciado mediante la sustitución de las fórmulas instanciadas con las metavariables, se muestra el cuaderno de resolución con el enunciado. El aspecto inicial que presenta el cuaderno de resolución cuando el alumno comienza a utilizar *MathEdu Solver* es el mostrado en la ilustración 3.27. Como observamos, el cuaderno de resolución es semejante al cuaderno de diseño que utilizó el profesor para diseñar el ejercicio. Se pone de manifiesto cómo se ha trasladado al alumno el modelo diseñado por el profesor, basándose en la programación mediante ejemplos.

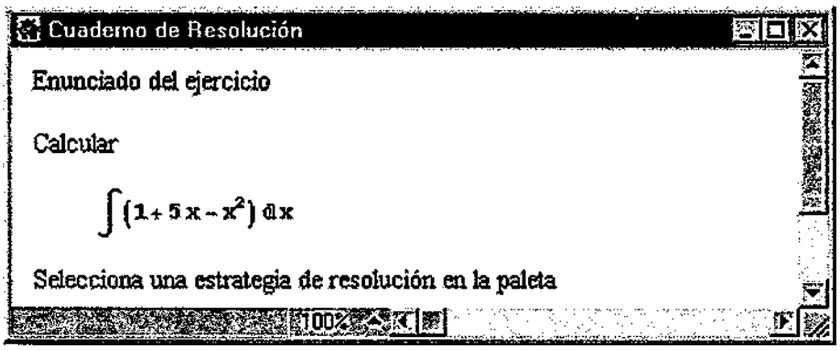

Ilustración 3.27. Cuaderno de resolución.

Como podemos observar en la ilustración anterior, el sistema le pide al alumno que seleccione una estrategia en la paleta para resolver el ejercicio. Esta parte de la resolución del ejercicio corresponde al acto de razonar sobre los datos y la estructura del mismo. Supone que el alumno tome su primera decisión, fundamental en cierto sentido, de la cual van a depender las acciones de resolución que se le van a presentar y que le han de llevar bien al éxito o al fracaso en la resolución del ejercicio. Este proceso relativo a la selección de estrategia va a ser objeto del siguiente apartado.

Para profundizar más en las estructuras de datos involucradas en los procesos que acabamos de describir se puede consultar el apartado G del apéndice A, en el que se muestra el seudocódigo de la función involucrada en el proceso de generación así como una esquematización del proceso sobre la estructura de datos de un tipo de ejercicio.

3.4.2. Selección de estrategias de resolución de ejercicios

Como acabamos de indicar vamos a profundizar en las cuestiones propias de la selección de estrategias. El presente apartado tratará con detalle los procesos seguidos en la misma.

Una vez que *MathEdu Solver* presenta al alumno el ejercicio generado aleatoriamente en el cuaderno de resolución, le solicita que seleccione una estrategia entre las estrategias susceptibles de ser utilizadas para resolver el ejercicio del tipo generado. Obviamente muchas de las estrategias que se le presentan no son válidas y carece de sentido utilizarlas, pero el sistema se las ofrece al alumno con el fin de que sea él el que discrimine entre el conjunto de las que se le muestran. La construcción de la paleta de selección de estrategias se hace con las etiquetas descriptoras correspondientes a las estrategias, a las que ya hicimos referencia al hablar de *MathEdu Designer*. Se muestran todas las estrategias del tipo de ejercicio seleccionado (en este caso *integración*). Por tanto, a priori el sistema no restringe en medida alguna el conjunto de posibles ramas del árbol de resolución del ejercicio. Según el ejercicio que se haya generado es posible que exista más de una estrategia susceptible de ser utilizada para resolver el mismo. *MathEdu Solver* actúa de forma absolutamente transparente. No aporta ningún tipo de información o ayuda y es el propio alumno, basándose en sus conocimientos y conceptos adquiridos, el que debe ser capaz de escoger entre alguna de las estrategias correctas.

Con la selección que el alumno realiza se produce uno de los procesos más importantes de *MathEdu Solver*. Este proceso consiste en la verificación de la estrategia seleccionada, evaluando si es o no apta para resolver el ejercicio propuesto. Llegados a este punto *MathEdu Solver* ya no usa para nada la información utilizada inicialmente para la generación del ejercicio. Lo único que "sabe" es que al alumno se le ha planteado un ejercicio de tipo *integración*, cuyo enunciado es

Ejercicio. Calcular

$$\int\left(1+5x-x^2\right)dx$$

Es interesante comentar aquí que existen ejercicios que por sus características pueden ser resueltos utilizando diferentes estrategias. Por ejemplo:

Ejercicio. Calcular

$$\int \frac{-2x+5}{\left(1+5x-x^2\right)} dx$$

Se puede resolver como integral inmediata *logarítmica* o bien como integral *racional*. De ahí la trascendencia del proceso que va a tener lugar a continuación de verificación de la estrategia escogida por el alumno, ya que a priori el sistema no cuenta con ninguna información al respecto.

Por tanto con la información del enunciado y con la estrategia seleccionada por el alumno, en este caso *polinómica*, el módulo decisor de estrategias de resolución debe analizar la selección y decidir si es o no apta para resolver el ejercicio. Este módulo está formado por un procedimiento que involucra la verificación de patrones. El seudocódigo de dicho procedimiento se muestra en el apéndice A, apartado H.

Existen múltiples consideraciones y aclaraciones que conviene hacer al respecto del procedimiento indicado. Con el fin de facilitar las explicaciones, hemos agrupado en los tres apartados que vienen a continuación la generalidad de las acciones que realiza el procedimiento. Además, antes de realizar comentario alguno relativo a cada una de las partes, conviene incidir en el hecho de que para tomar la decisión de si la estrategia seleccionada es o no apta para resolver el ejercicio planteado, sólo son necesarios los parámetros siguientes:

- El tipo de ejercicio en resolución,
- La estrategia seleccionada por el alumno,
- Las fórmulas contenidas en el enunciado.

I. Obtención de datos.

En esta parte se encuentran todas las funciones encaminadas a seleccionar los datos relativos al tipo de ejercicio en resolución, a la estrategia seleccionada por el alumno y a las metavariables de dicha estrategia. Además se liberan de valor las metavariables, considerando únicamente los símbolos de las mismas.

II. Generación de patrones.

En este grupo se efectúan dos operaciones fundamentales

i. Se seleccionan los patrones de las fórmulas del enunciado

Como ya se explicó en el epígrafe 3.3, los patrones estructurales de las fórmulas del enunciado están formados por reglas en las que la parte izquierda representa el símbolo asociado a la fórmula y la parte derecha está formada por la representación de la fórmula propiamente dicha, la cual está expresada en función de los símbolos de las metavariables. Para comprobar si la selección de estrategia es correcta necesitamos la parte derecha de las reglas que determinan la estructura abstracta de las fórmulas del enunciado. A partir de los consecuentes de las reglas se van a obtener patrones condicionados de dichas fórmulas. Con el fin de aclarar esto vamos a analizar en qué consiste este proceso en el caso del ejemplo que nos ocupa. La parte derecha de la regla correspondiente a la fórmula del enunciado es

Integrar(*polinomio_*, *x*) [1]

Vemos que la expresión [1] es función del símbolo de metavariable *polinomio*. En el apartado siguiente vamos a convertir esta expresión en un patrón condicionado, en el que el símbolo *polinomio* aparecerá condicionado por el condicionante correspondiente a la metavariable.

ii. Se construyen las expresiones simbólicas condicionadas de los patrones seleccionados

En esta fase a partir del patrón obtenido en la etapa anterior y de la lista de metavariables obtenida en el paso I, se obtiene el mismo patrón en el que las metavariables aparecen asociadas a sus condicionantes, dando lugar a un patrón condicionado. En el caso del ejemplo, la expresión [1] se transforma en la siguiente, similar a la anterior pero en la que la metavariable *polinomio* ya aparece ligada a su condicionante, que en este caso requiere que dicha expresión esté representada por un polinomio.

Integrar(*polinomio_?PolynomialQ*, *x*) [2]

con el fin de poder verificar si la expresión de la fórmula del enunciado es compatible con el patrón condicionado asociado a la selección del alumno.

En el caso de que existan distintos casos para una misma estrategia se obtienen tantos patrones condicionados como casos haya. Por ejemplo en una integral por partes pueden generarse los patrones condicionados siguientes, cada uno de los cuales hace referencia a unos de los casos de ejercicios resolubles como integral por partes.

Integrar(*u_?PolynomialQ* ·*v_?TrigQ, x*)
Integrar(*u_?PolynomialQ* ·*v_?ExpQ, x*)
Integrar(*u_?PolynomialQ* ·*v_?LogQ, x*)
Integrar(*u_?PolynomialQ* ·*v_?ExpDecQ, x*)

III. Decisión.

En este paso se efectúa la comparación entre las expresiones de las fórmulas en el enunciado y los correspondientes patrones condicionados asociados a la estrategia escogida por el alumno. La comparación es posible gracias a la función **MatchQ** que se explicó en 2.5.2. Recordemos que esta función toma como argumentos una expresión y un patrón condicionado y devuelve Verdadero o Falso dependiendo de que la expresión en cuestión cumpla o no el patrón. En el caso del ejemplo que nos ocupa, el resultado de la operación

$$\textbf{MatchQ}\left(\int\left(1+5x-x^2\right)dx,\ \text{Integrar}(\textit{polinomio\_?PolynomialQ}, x)\right) \qquad [3]$$

es Verdadero puesto que la fórmula

$$\int\left(1+5x-x^2\right)dx$$

tiene por expresión simbólica en el lenguaje simbólico de *Mathematica*

Integrar(Sumar(1, Multiplicar(5, *x*), Multiplicar (-1, Potencia(*x*, 2))), *x*) [4]

de modo que la función **MatchQ** compara el primer con el segundo parámetro de [3], siendo ambas expresiones iguales salvo en la parte correspondiente al integrando. Por tanto, al final la comparación se reduce a comprobar si

Sumar(1, Multiplicar(5, *x*), Multiplicar (-1, Potencia(*x*, 2)))

verifica el patrón condicionado *_?PolynomialQ* (el nombre del patrón, *polinomio* en este caso, es irrelevante). Obviamente el resultado de la comparación debe ser Verdadero. En definitiva, en este caso la decisión para la selección de estrategia realizada por el alumno, ha sido compatible con el ejercicio planteado. Supongamos por un momento que la selección de la estrategia que escoge el alumno hubiese sido *por partes*. Uno de los posibles patrones condicionados de dicha estrategia es, como acabamos de ver,

Integrar(Multiplicar(*u_?PolynomialQ, v_?TrigQ*), *x*)

En este caso, la comparación

MatchQ(
$\int(1+5x-x^2)\,dx$,
Integrar(Multiplicar (*polinomio_?PolynomialQ, simpleTrig_?TrigQ*), *x*))

debe producir como resultado Falso, puesto que aunque en el integrando del patrón condicionado aparece descrito un polinomio, también aparece multiplicando a este una función trigonométrica. Dado que en la expresión simbólica de la fórmula del ejercicio tal función trigonométrica no está, el resultado de la comparación semántica resultaría ser Falso.

Una vez tomada la decisión y puesto que en el paso I hemos considerado las metavariables sin su valor asociado es necesario que, una vez hecha la comparación de patrones y antes de abandonar el procedimiento, se vuelvan a ligar las metavariables con sus valores para continuar con el ejercicio. Para concluir, se comprueba si la decisión tomada es adecuada o no. En el primer caso se continúa con el ejercicio adelante y, caso de no serlo, se envía al alumno un mensaje de error y se le permite volver a escoger otra estrategia.

No debemos concluir este apartado sin hacer referencia a otra cuestión importante asociada a la selección de estrategia. Nos referimos a la posibilidad de que un ejercicio sea resoluble mediante dos o más estrategias diferentes. Esta es una circunstancia muy habitual en el razonamiento matemático. La existencia de caminos diferentes para resolver un mismo ejercicio, cuestión esta que dificulta enormemente la sistematización de los procesos de razonamiento y resolución y que pensamos que ha sido, en gran medida, una de las causas de que hasta el presente no se haya abordado con decisión la resolución interactiva de ejercicios y problemas en disciplinas científicas en general y en Matemáticas en particular.

Para encontrarnos situaciones en las que la circunstancia comentada anteriormente se da no es necesario recurrir a complejos ejercicios de integración, ni de ecuaciones diferenciales, ni similares. Basta con un simple ejercicio de aritmética con las operaciones básicas suma y producto. Veámoslo con el siguiente ejemplo.

Ejercicio. Calcular $\frac{2}{3}\left(\frac{5}{2}+\frac{1}{5}\right)$

Este es un ejercicio básico de aritmética de fracciones. Pero pensemos cómo puede resolverse. Una posibilidad es efectuar la suma y el resultado de la misma multiplicarlo por

el factor $\frac{2}{3}$. Tampoco sería incorrecto aplicar la propiedad distributiva y multiplicar el primer factor por cada uno de los sumandos en el segundo factor, sumando posteriormente ambos resultados parciales.

Vemos, pues, que hay (al menos) dos formas distintas de resolver un ejercicio con estas características. Sería deseable, por tanto, que el alumno dispusiera de al menos esas dos estrategias para resolverlo. Pues bien, tal circunstancia es perfectamente posible realizarla con *MathEdu*.

Desde el punto de vista de las estrategias posibles (llamémoslas *simplificar* y *distributiva*), ambas dan lugar al mismo patrón condicionado puesto que comparten el mismo patrón estructural de fórmula y las mismas metavariables. Sin embargo, lo que distingue a una estrategia de otra en este caso son las acciones (recordemos la definición de estrategia en el epígrafe 3.3.3.2). En el caso de la estrategia ***distributiva*** las acciones que se han de realizar se resumen en:

- *Efectúa el producto del factor común por el primer sumando.*
- *Efectúa el producto del factor común por el segundo sumando.*
- *Suma los resultados parciales de las operaciones anteriores.*

En el caso de la estrategia ***simplificar*** las acciones a realizar son

- *Resuelve la suma del segundo factor en el enunciado.*
- *Efectúa el producto del factor común por el resultado obtenido de la suma.*

En este segundo caso el diseño de las acciones de resolución se ha basado en la acción *resolución de ejercicio*, invocando la resolución de la suma correspondiente al segundo factor para, posteriormente, multiplicar dicho resultado por el primer *factor*. En consecuencia, comprobamos que es posible reproducir el razonamiento humano para resolver este ejercicio sencillo de cálculo abordándolo desde los dos puntos de vista posibles, facilitando al alumno la posibilidad de escoger entre ambas estrategias, siendo ambas igualmente válidas. *Solver* acepta cualquiera de las dos selecciones de estrategia, ya que ambas se adaptan al patrón semántico del ejercicio. En este caso, como ya hemos indicado antes, las diferencias entre ambas estrategias vienen dadas por las acciones asociadas a cada una de ellas.

En el apartado I del apéndice A se detalla la estructura de datos, la cual contempla ambas estrategias, asociada para un ejercicio con estas características.

Un hecho importante a resaltar es la presencia en la acción de *resolución de ejercicio* de parámetros de entrada y salida. A diferencia de lo que sucede en la generación aleatoria de ejercicios (apartado 3.4.1), la función encargada de la resolución de los ejercicios distingue los casos en los que no existen datos de entrada ni salida (en cuyo caso procede a la generación aleatoria del enunciado), de los casos en los que sí se invoca dicha función con datos tanto en la entrada como en la salida, como en el caso anterior. Un segundo efecto notable asociado a esta circunstancia es que la resolución de este subproblema, el que corresponde a la obtención del factor suma, se efectúa en otro cuaderno de resolución. Naturalmente, al tratarse de otro ejercicio, se va a pedir al alumno nuevamente que seleccione una estrategia de resolución, una explicación conceptual asociada a dicha estrategia y que resuelva las acciones que el sistema le vaya presentando.

Finalmente no queremos avanzar al siguiente apartado sin mostrar con otro ejemplo la relevancia que puede tener una apropiada selección de estrategia de resolución. El siguiente ejemplo puede resolverse utilizando, al menos, tres estrategias distintas de resolución

$$\int \frac{2x+1}{x^2+x} dx$$

Dichas estrategias son:

1. Es una integral *inmediata*. Observemos que el numerador es la derivada del denominador y, en consecuencia, la función primitiva corresponde al logaritmo neperiano del polinomio del denominador.

2. Se puede resolver por *cambio de variable*. Si denominamos a $x^2+x=t$ entonces $(2x+1)dx=dt$ y la integral se convierte en

   $$\int \frac{dt}{t}$$

   que reduce el problema al caso anterior.

3. Finalmente la integral se puede resolver como *racional*. Tendríamos que obtener la descomposición en fracciones simples del integrando quedando

   $$\int \frac{2x+1}{x^2+x} dx = \int \frac{1}{x} dx + \int \frac{1}{x+1} dx$$

   las cuales son, a su vez, integrales inmediatas del tipo logarítmico descrito en el punto 1.

Con este ejemplo se pone de manifiesto la relevancia de trabajar en un nivel simbólico con las metavariables carentes de valor (tal y como se ha indicado en el paso I anterior), porque ello nos permite tomar decisiones desde un punto de vista estructural, independientemente del valor que en un ejercicio concreto tengan las metavariables.

3.4.3. Selección de la descripción asociada a la estrategia

Toda estrategia debe llevar asociada una descripción de las tareas básicas que conlleva la misma. Dicha descripción la debe haber declarado el profesor en el proceso de diseño del ejercicio. Con el fin de comprobar, en cierta medida, si la selección del alumno ha sido hecha de forma razonada o al azar, *MathEdu Solver* va a hacer uso de la información textual de que se dispone en dichas descripciones para someter al alumno a una primera prueba sobre la decisión tomada.

En concreto, *Solver* va a presentar al alumno una paleta con todas las posibles explicaciones correspondientes a las estrategias del tipo de ejercicio que se esté resolviendo. Las opciones de esta paleta están ordenadas de forma aleatoria con el fin de que el alumno no tenga una referencia espacial de la estrategia que ha escogido en situaciones similares anteriores. Sólo una de las explicaciones es compatible con la estrategia que ha escogido y *Solver* continuará con la resolución del ejercicio únicamente en el caso de que el alumno seleccione la explicación correcta. En el caso contrario se le envía un mensaje de error recomendándole que repase la teoría correspondiente al tema en cuestión.

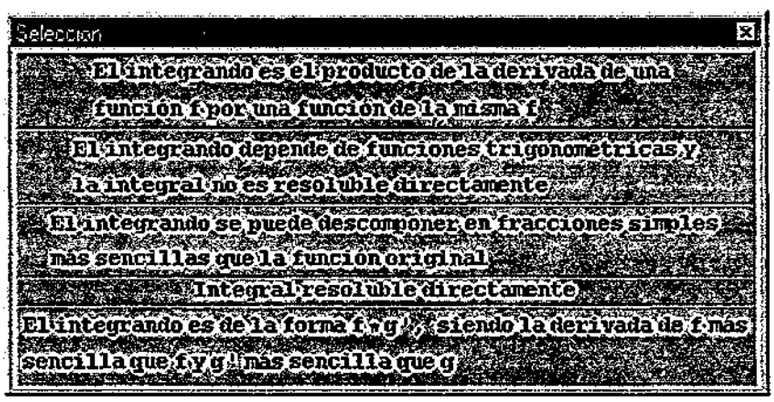

Ilustración 3.28. Descripción de las estrategias de resolución de un ejercicio de integración.

La ilustración 3.28 muestra una de estas paletas correspondiente a las descripciones de las estrategias asociadas al tipo de ejercicio *integración*. Evidentemente es una forma muy superficial de evaluación. Pero no es menos cierto que una decisión tan aparentemente simple como seleccionar la estrategia a través de la cual el alumno cree que puede resolver

el ejercicio conlleva haber efectuado un análisis pormenorizado de la estructura que observa en el mismo, identificándolo con alguno conocido. El alumno debe efectuar esta operación de un modo consciente, procurando conocer los conceptos relativos del tema al cual corresponde el ejercicio. Una selección aleatoria de la estrategia de resolución le llevará, casi con total seguridad, bien a una resolución errónea o bien a no ser capaz de continuar con la resolución. En cualquiera de los dos casos, el acto consciente del aprendizaje quedará completamente desvirtuado y anulado ante un desconocimiento teórico insuficiente por parte del alumno.

3.4.4. Acciones del alumno

El último módulo del motor de resolución de *MathEdu Solver* corresponde al gestor de las acciones del alumno. Desde el primer momento en que comenzamos a diseñar cómo debería ser un sistema de las características de *MathEdu* siempre hemos tenido la idea, más o menos clara, de que la resolución de un ejercicio vista en su totalidad es similar a ver los fotogramas de una película, donde cada fotograma representaría una acción de la resolución del ejercicio y la película entera representaría la resolución completa del ejercicio. La característica fundamental de *MathEdu* consiste en representar películas abstractas, válidas para nuevos problemas, en una sola estructura de datos que en cada caso se puede convertir en una película específica interpretable por el motor de resolución.

Esta concepción filmográfica de la forma en que se puede abordar la resolución de un ejercicio de Matemáticas nos ha sido de gran utilidad a la hora de pensar en el modo en que debíamos diseñar las acciones que el alumno debe ejecutar y, por tanto, en la interacción hombre-máquina. La mayor dificultad que hemos encontrado al plantear esta forma de razonar, considerando que el alumno debía ir encadenando las acciones que el profesor había especificado en la fase de diseño, se debió a que *Mathematica* no captura eventos del sistema. Es decir, no existe en *Mathematica* la posibilidad de crear un bucle que detecte, por ejemplo, el cambio de valor de una variable, o que el usuario ha escrito algo en un campo de introducción de datos, o que se ha abierto un menú o se ha pulsado un botón. Estos aspectos básicos en la programación de aplicaciones con interacción hombre-máquina, como es el caso de *MathEdu*, resultan una carencia significativa en un sistema de las características de *Mathematica*. Para suplir esta carencia pensamos en un esquema más rudimentario pero que en definitiva ha resultado igualmente eficaz. La idea básica es que todos los elementos de la interfaz que utilizamos, salvo los cuadernos, que son meros expositores de información, disponen de algún elemento activo (uno o varios botones). Dichos elementos son los que nos permiten encadenar la sucesión de tareas que tanto en la fase de diseño como en la de resolución han de ejecutar los usuarios del sistema.

En el caso de la gestión de acciones en *Solver*, que es el módulo que ahora nos ocupa, dicha gestión la realiza la función **hacerAcciones**, cuyo único cometido es identificar el tipo de la acción (su *cabecera*) y traducirla a una función ejecutable por *Mathematica*. El

seudocódigo de dicha función se detalla, como es habitual, en el apartado J del apéndice A. Su funcionamiento es muy simple: toma siempre la primera acción de la lista de acciones (la cual actúa como una cola *First-In First-Out*) que se le pasa como parámetro y, dependiendo del tipo que sea la acción (lo cual lo indica la cabecera de la misma), se reemplaza dicha cabecera por otra correspondiente a una función del módulo *Solver* la cual es ejecutada de forma automática por el *núcleo* de *Mathematica*, dando lugar al diálogo de la acción correspondiente que debe realizar el alumno.

Como se puede apreciar todas las acciones tienen como último parámetro la cola de acciones. Cada acción porta la información del resto de acciones que aún quedan por ejecutar y, una vez que ha concluido ella misma, invoca nuevamente a la función **hacerAcciones** con la lista del resto de acciones que deben ejecutarse. Gráficamente lo esquematizamos en la figura 3.6 siguiente. Es importante comprobar cómo todas las acciones siempre invocan nuevamente a la función **hacerAcciones**, salvo en el caso de la función que invoca a **resolverProblema** para iniciar la resolución de un subproblema.

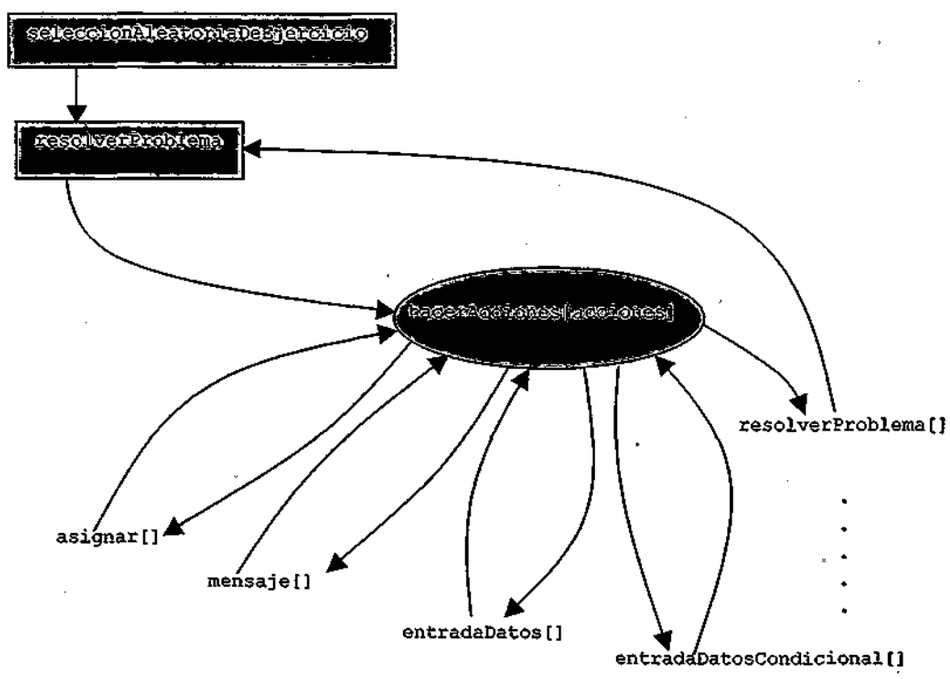

Figura 3.6. Representación esquemática del bucle `hacerAcciones`.

En el apartado 3.3.3.2.2, hemos descrito con detalle cómo el profesor diseña las acciones que el alumno va a realizar posteriormente en la fase de resolución interactiva. Remitimos al lector a aquel apartado con el fin de recordar cualquier aspecto que se considere oportuno al respecto de los diferentes tipos de acciones. Únicamente queremos

presentar el diálogo utilizado para las acciones de entrada de datos, el cual resulta fundamental en el diálogo alumno/sistema.

Diálogo para las acciones de entrada de datos

Las acciones de *entrada de datos* utilizan todas ellas el diálogo de la ilustración 3.29, a través del cual el alumno puede introducir en el sistema la expresión simbólica o el dato que se le solicita. El alumno dispone de la paleta de símbolos habitual de *Mathematica*, pudiendo hacer uso de cualquiera de ellos si lo estima oportuno.

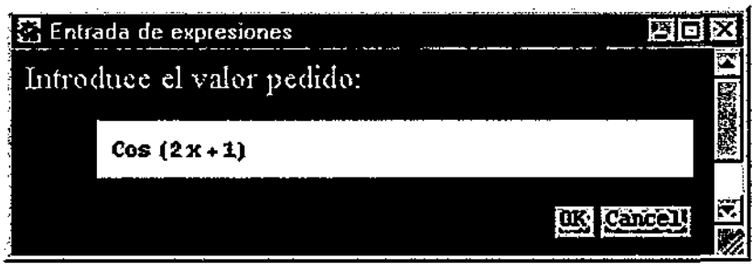

Ilustración 3.29. Diálogo de la acción de *entrada de datos*.

Conviene recordar aquí que en las acciones de entrada de datos con patrón las condiciones que puede especificar el diseñador son de dos tipos. Por un lado puede especificar que la condición sea del tipo *predicado* (*PolynomialQ*, *NumberQ*, *IntegerQ*, etc.). Por otro lado puede especificar patrones condicionados de la expresión. Por ejemplo son condiciones de la forma

$$\int Sen(\_?PolynomialQ)\,dx$$

la cual exige que la expresión que teclee el alumno debe ser una integral cuyo integrando esté formado por la función seno aplicada a una expresión polinómica. En este patrón semántico el profesor exige condiciones tanto sobre la estructura de la expresión (debe ser una integral de un seno) como sobre el contenido del argumento de dicha función trigonométrica (debe ser un polinomio). Es en este aspecto de la programación con *MathEdu* en la que pensamos que es necesario disponer de unos mínimos conocimientos de *Mathematica* y sin duda somos conscientes de que algunas de las mejoras que han de realizarse sobre esta versión inicial de la aplicación, deben ir en la dirección de mejorar la interfaz con el profesor para facilitarle este tipo de tareas de programación.

La posibilidad de poder comprobar semánticamente expresiones confiere a *MathEdu* una característica que le aproxima en cierta medida al modo de razonamiento humano, al ser capaz de considerar como válidas expresiones tecleadas por el estudiante que, aunque estructuralmente difieren de las previstas en su día por el profesor durante el diseño, semánticamente tienen el mismo valor. En el apartado K del apéndice A se puede encontrar información más detallada sobre el proceso de evaluación de expresiones.

3.4.5. Conclusiones

La importancia del conjunto de procesos que acabamos de explicar relacionados con las acciones de resolución de un ejercicio radica en el hecho de que gracias a la existencia de la acción *resolver* se consigue establecer para un mismo ejercicio distintas alternativas de resolución del mismo. La propia naturaleza de las acciones de resolución de un ejercicio hacen que esta sea en general excesivamente lineal. Los ejercicios de cálculo que se pueden diseñar y resolver con *MathEdu* son algorítmicos. Esta afirmación la hacemos en el sentido de que siempre han de ejecutarse las mismas acciones para resolver un ejercicio de un mismo tipo utilizando una misma estrategia. La importancia de la cuestión radica en, a nuestro juicio, dos hechos

i. Que se pueden sistematizar las acciones de resolución, ayudando al alumno a adquirir destreza en las mismas (este es el aspecto algorítmico a que nos acabamos de referir)

ii. Que disponemos de una forma de diálogo sencillo con el alumno, logrando una interacción hombre-máquina que hasta el momento no se había logrado en términos de resolución de ejercicios de Matemáticas.

La existencia de la acción *resolución de ejercicio* posibilita la utilización de modos alternativos de resolución de partes de un ejercicio incrementando los beneficios obtenidos con las restantes acciones de *MathEdu*. Esto se concreta invocando la resolución de subproblemas que subyacen en la resolución del ejercicio principal. El matemático tiende siempre a parcelar su actividad y, en el caso de la resolución de ejercicios que involucran cálculo simbólico, esta circunstancia también está presente. Por tanto este mecanismo de *MathEdu* es similar al modo en que razona una persona.

3.5. MathTrainer

3.5.1. Descripción y aspectos generales de la aplicación.

Comenzamos el epígrafe final del presente capítulo en el cual vamos a describir las características propias de *MathTrainer* (Díez, 2000; Díez, 2001), aplicación integrada en *MathEdu* la cual se ha obtenido a partir de *MathEdu Solver*. *MathTrainer* surgió de forma natural a la vista de las capacidades que nos ofrecía *Solver* con el fin de tratar de proporcionar explicaciones al alumno acerca de un ejercicio que se le hubiese planteado y, o bien no supiese como resolverlo, o bien al resolverlo se encontrara con dificultades inesperadas e inclusive en el caso de que el alumno deseara saber cómo resolver un ejercicio que él planteara a *MathEdu*. La forma en que trabaja *MathTrainer* puede considerarse próxima a campos propios de la Inteligencia Artificial puesto que partiendo del patrón estructural de un ejercicio determina la estrategia o estrategias válidas para resolverlo y, a partir de dichas estrategias encadena las acciones asociadas, realizando los cálculos oportunos en cada momento y mostrando al alumno paso a paso cómo se debe resolver el ejercicio.

En las páginas siguientes vamos a mostrar las distintas formas en que puede ser usado *MathTrainer* tanto por el alumno como por el profesor, así como la interfaz de la aplicación.

El Alumno

El alumno que sigue de forma interactiva un curso de cálculo puede usar *MathTrainer* en el momento que lo desee. Un curso consiste en una secuencia de capítulos que pueden incluir teoría, ejemplos y ejercicios para ser resueltos. Durante su estudio, el alumno generalmente primero lee la teoría y los ejemplos y posteriormente dispone de dos alternativas:

1. Puede intentar los ejercicios que dinámicamente genera *MathEdu Solver*. Los ejercicios son generados a partir de la información del tipo de ejercicio que desea resolver el alumno, correspondiente a alguno de los temas tratados dentro de un curso.
2. Puede estudiar cómo resolver ejercicios de un modo guiado, paso a paso. A este respecto diremos que los ejercicios concretos que se le van a presentar al alumno no están predefinidos. El alumno escoge el tema y el tipo de ejercicio es seleccionado de forma aleatoria por el sistema.

En el segundo caso el alumno puede escoger entre dos modos de selección de ejercicios posibles:

1. Selección aleatoria del tipo de ejercicio.

2. Introducción del ejercicio por teclado

En el primer caso el sistema se encarga de generar un enunciado de forma aleatoria y, a partir del mismo, ilustrar la resolución guiada del mismo.

En el segundo caso el alumno dispone de un diálogo para teclear la expresión correspondiente al tipo de ejercicio que haya seleccionado. Es decir, el alumno le indica al sistema el tipo de ejercicio que desea resolver, por ejemplo ecuaciones diferenciales ordinarias, derivación o integración. A partir de esta información el sistema construye un diálogo que representa el enunciado correspondiente al tipo seleccionado. En dicho diálogo los campos correspondientes a las fórmulas presentes en el enunciado son libres y el alumno puede teclear cualquier expresión que desee. Podemos ver un diálogo de estas características en la ilustración 3.30.

En ambos casos el alumno dispone de un botón para avanzar paso a paso. Cada vez que pulsa el botón el sistema pasa a la siguiente acción de la lista de acciones que debe ejecutar, la realiza y queda en espera hasta que el alumno vuelva a solicitar el siguiente paso.

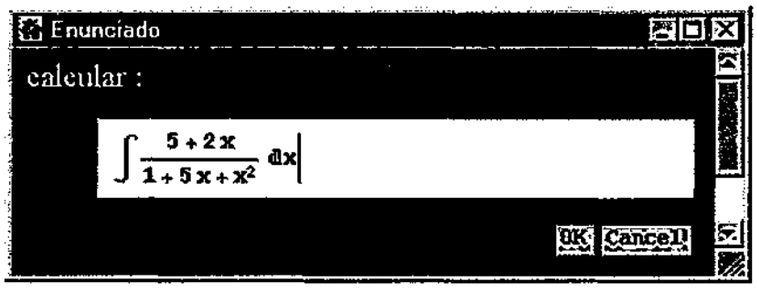

Ilustración 3.30. Diálogo de introducción de fórmulas para un tipo de ejercicio.

Por tanto, desde el punto de vista del alumno, *MathTrainer* es exclusivamente un programa que le permite visualizar secuencialmente los pasos necesarios que debe dar para resolver un ejercicio, junto con los resultados que debe obtener en cada paso.

En consecuencia una vez que el alumno ha estudiado por ejemplo cómo es la resolución de una ecuación diferencial lineal como

$$y' + y = 1$$

puede decidir que el sistema le muestre cómo resolver ecuaciones de la forma

$$y' + y/x = x^2$$

MathTrainer entonces le dará al alumno una explicación, paso a paso, de cómo debe realizarse la resolución de este tipo de ejercicios. Para que el sistema pueda dar dicha explicación es necesario que sea capaz de identificar las expresiones que teclea el alumno entre aquellos patrones de que dispone en la estructura de datos del curso. En caso de que el ejercicio propuesto no fuese apropiado, en el sentido de que el sistema es incapaz de reconocer el patrón asociado al mismo entre el conjunto de patrones asociados a cada tipo de ejercicio, el sistema se lo comunicará al alumno, el cual podrá revisar el tipo de datos que ha utilizado para plantear el ejercicio al sistema y a qué se debe el que este no sea capaz de mostrarle la resolución.

Otra de las formas en que puede integrarse *MathTrainer* con *MathEdu Solver* consiste en proporcionar al alumno explicaciones de cómo proceder en caso de cometer errores de forma reiterada en los cálculos que debe ir realizando de forma interactiva a instancias de *MathEdu Solver* o bien en caso de que sea el propio alumno el que solicite la ayuda al no saber cómo realizar un paso. En la versión inicial de *MathEdu*, tal y como fue diseñado en un principio, este tipo de ayuda sólo era posible darla en caso de que el profesor hubiera reparado en la posibilidad de que en determinadas circunstancias podría ser necesaria determinada ayuda y la hubiera planificado "a mano". Esta era una limitación importante de *MathEdu* dado que puede darse la circunstancia de tener que proporcionar ayudas en cada paso del trabajo desempeñado por el alumno, en cuyo caso la cantidad de trabajo adicional que debía realizar el diseñador para proporcionar explicaciones adicionales podía ser desmesurada.

El profesor

Una cuestión importante es en qué medida afecta el uso de *MathTrainer* al profesor. En este sentido hemos de decir que el proceso de diseño de cursos por el profesor que está a cargo de los mismos mediante el uso de la herramienta de autor *MathEdu Designer* posibilita que el uso de *MathTrainer* sea una actividad absolutamente transparente para el mismo. Cuando el profesor está diseñando un ejercicio debe concentrarse exclusivamente en los aspectos relativos al diseño, qué estrategia de resolución va a declarar, qué partes de las fórmulas va a generalizar, qué acciones va a definir, etc. En ningún momento debe pensar en los aspectos propios a la resolución guiada porque es el propio *MathTrainer* el que haciendo uso de la información del diseño se encarga de efectuar la resolución guiada paso a paso.

Radica aquí el valor añadido que esta aplicación ha proporcionado al sistema original. El profesor obtiene "por el mismo precio" de su trabajo de diseño de ejercicios, la explicación de los mismos. Sólo tiene que preocuparse, como ya hemos visto en partes anteriores del presente capítulo, en los procedimientos algorítmicos que desea diseñar a fin de que el alumno se ejercite en los mismos. Por tanto el proceso de diseño puede considerarse como el de definición de las acciones que tanto el sistema como el alumno

deben realizar en tiempo de resolución. Recordemos que *MathEdu Designer* almacena una abstracción de las acciones que deben ser utilizadas *a-posteriori*. Cuando *MathTrainer* muestra al alumno cómo resolver un ejercicio, bien generado por el propio *MathTrainer* o puesto por el alumno, se limita a reproducir exactamente las mismas acciones que especificó el profesor una vez puestas en el contexto correspondiente y realiza por sí mismo los cálculos oportunos que deben ejecutarse para cada una de ellas, mostrando los resultados al alumno en el correspondiente cuaderno de resolución.

Por otro lado, cuando el alumno está resolviendo un ejercicio, *MathEdu Solver* verifica cada expresión que el alumno introduce, contrastando su compatibilidad con el patrón esperado para dicha expresión, el cual fue especificado en el momento de diseño por el profesor. Este proceso de validación utiliza la potencia de los mecanismos de comparación de patrones de que dispone *Mathematica*, los cuales posibilitan la definición de patrones de estructuras complejas de expresiones matemáticas. Estos mismos mecanismos también son usados por *MathTrainer* en el mismo modo en que se usan en *Solver*. Del mismo modo que *Solver* determina las estrategias que son válidas para resolver un ejercicio, *MathTrainer* también realiza este proceso y le pide al alumno que escoja, caso de haber más de una, la que él desee para mostrar la resolución guiada. Una vez que el alumno escoge la estrategia que desea explorar, el sistema determina su conjunto de acciones asociadas y comienza el proceso de demostración de la resolución.

La interfaz

El proceso de demostración de la resolución utiliza una interfaz equivalente a la que utiliza *MathEdu Solver* con la diferencia que en vez de ir mostrando ventanas de diálogo al alumno para que introduzca las expresiones que se le piden, es el propio sistema el que va mostrando las expresiones que el alumno deberá introducir en una hipotética sesión de resolución interactiva.

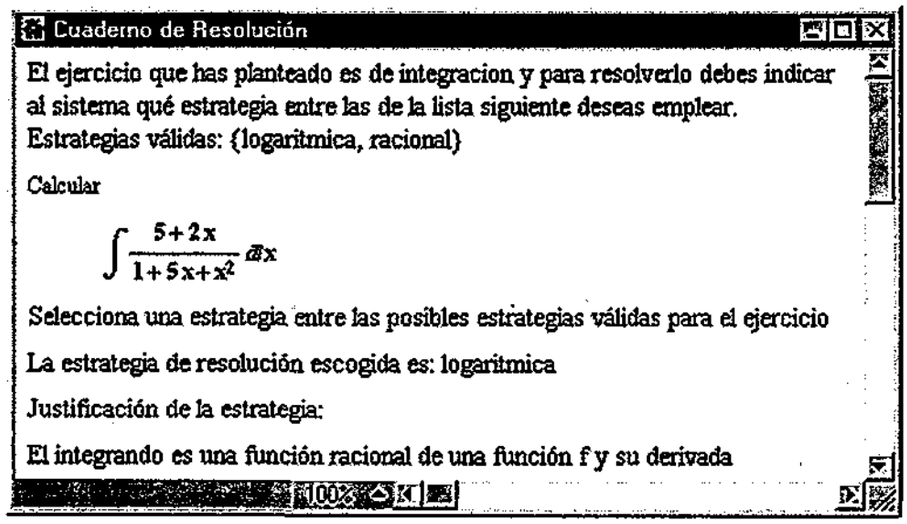

Ilustración 3.31. Ejemplo de cuaderno de resolución guiada

En la ilustración 3.31 se muestra un ejemplo de cuaderno de resolución guiada de un ejercicio de integración que se resuelve mediante una integral de tipo logarítmico. De esta forma el alumno puede ir leyendo en el cuaderno de resolución guiada los mensajes de las diferentes acciones que debe realizar y los resultados de la ejecución de dichas acciones. También encontrará en el cuaderno los mensajes del sistema relativos a la selección inicial de estrategia y de su justificación. Esta interfaz está adaptada para mostrar en cada ejecución los datos concretos del ejercicio que se está resolviendo, de forma que la explicación final del ejercicio es semejante a la que se podría encontrar en un libro de texto.

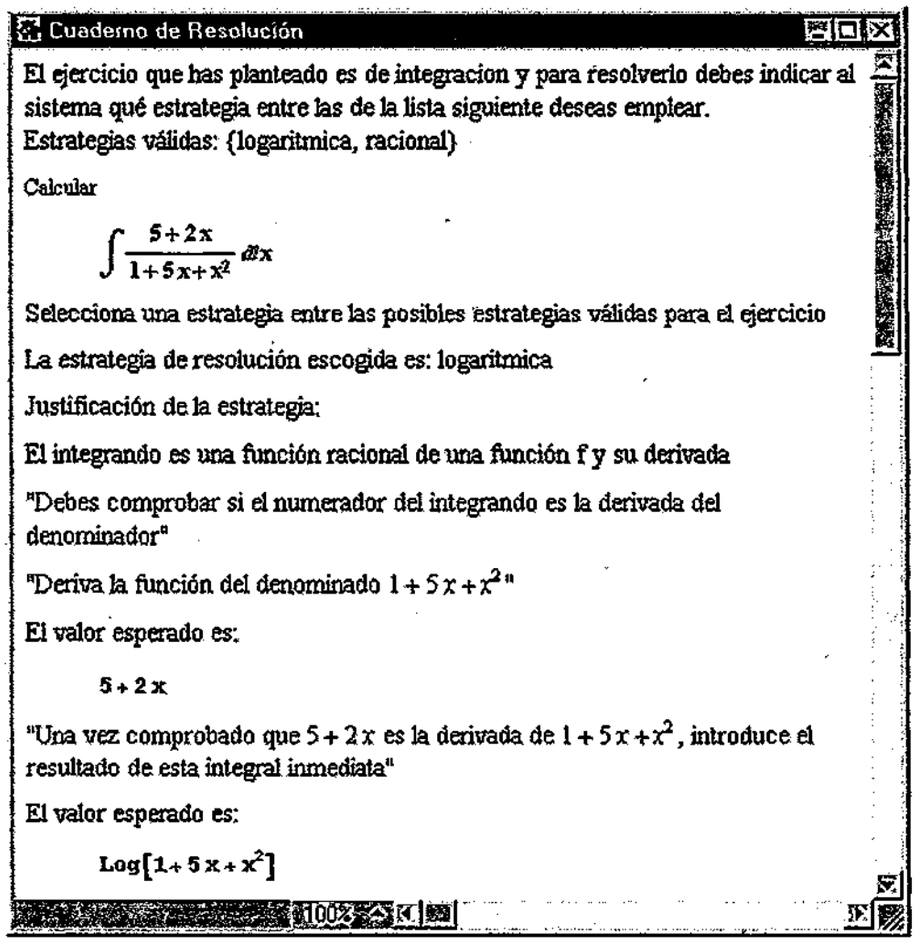

Ilustración 3.32. Continuación del cuaderno anterior.

La ilustración 3.32 muestra la continuación de este mismo ejemplo. En la continuación podemos apreciar como hay, en este caso, dos tipos distintos de acciones. Por un lado tenemos una acción del tipo *mostrar mensaje*, la primera que aparece en el cuaderno. Sin embargo la segunda acción ya corresponde a un tipo de acción de entrada de datos (probablemente una acción de *entrada de expresiones*), en la cual se pide al alumno que derive la expresión que se muestra, un polinomio de grado 2. En negro se muestran los resultados

de dicha operación. Obviamente en este caso el resultado de la operación no ha sido introducido por el alumno, sino que ha sido la propia aplicación, *MathTrainer*, haciendo uso del núcleo de *Mathematica*, la que ha pedido a éste que obtenga la derivada pedida y, acto seguido, la muestra en el cuaderno. Una situación similar se reproduce en la última acción que se muestra.

La forma en que actúa *MathTrainer* es similar a la que lo hace *MathEdu Solver* de tal modo que las acciones de *resolución de ejercicio* provocan la apertura de un nuevo cuaderno de resolución en otra ventana y todos los procedimientos derivados de la resolución de un ejercicio nuevo (del mismo tipo o de otro distinto). Se vuelven a analizar las estrategias válidas, se le muestran al alumno para que escoja la que desee, se justifica la estrategia y se encadenan sus acciones hasta concluir el subproblema, momento en el que se regresa a la resolución del ejercicio original.

3.5.2. Conclusiones.

Como hemos podido comprobar, *MathTrainer* complementa de manera idónea la idea original de *MathEdu* de interactuar con el alumno en la resolución interactiva de ejercicios. Y más aún si tenemos en cuenta que el esfuerzo que esto supone al profesor es nulo, la incorporación de una aplicación con estas características al sistema parece, a todas luces, óptima.

MathTrainer incorpora una nueva funcionalidad que no estaba disponible en la concepción inicial del sistema. Esta nueva funcionalidad hace más sencilla la definición de cursos de materias que involucran cálculo simbólico y que permiten al alumno aprender más rápido y mejor los diferentes métodos de resolución de los ejercicios relacionados con tales materias. En este sentido, el objetivo principal es hacer sentir al alumno que el sistema es un colaborador suyo durante su proceso de aprendizaje, consiguiendo que se familiarice y que confíe en él. Para lograr esto es indudable que, además de utilizar interfaces suficientemente gratas, próximas al alumno, claras y concisas, es conveniente incorporar funcionalidades propias de los tutores inteligentes, fundamentalmente un modelo del alumno, que recoja las vicisitudes personales de cada uno individualmente, conociendo sus virtudes y deficiencias y tratando de dar ayuda en aquellos momentos en los que exista cierta verosimilitud de que el alumno la va a necesitar. En este sentido volveremos a tratar estos aspectos propios de la concepción del sistema cuando planteemos las conclusiones finales de la memoria.

Parte III
Resultados y Conclusiones

Capítulo 4.

PRUEBAS DE DISEÑO Y RESOLUCIÓN CON *MathEdu*

En el presente capítulo vamos a exponer en qué han consistido las pruebas que hemos realizado con *MathEdu* al objeto de probar sus características de funcionamiento en lo referente tanto a la fase de diseño como a la de resolución, tanto interactiva como guiada.

Como ya explicamos al inicio de la tesis, el propósito del trabajo era el de desarrollar una tecnología que permitiera al profesor, de una parte, diseñar los procesos básicos que describan ejercicios de cálculo para que el alumno, de otra parte, pudiese resolverlos de forma interactiva. En definitiva el objetivo de la tesis era, fundamentalmente, tecnológico. Por consiguiente, a la hora de plantear las pruebas del sistema no hemos querido probar la capacidad de *MathEdu* para enseñar dado que el planteamiento general de la tesis no es sobre didáctica. Esto no quiere decir, claro está, que no deban hacerse pruebas reales con profesores y alumnos que evalúen la utilidad de la herramienta desde el punto de vista de los usuarios finales. Pero como ya hemos expuesto en la memoria son necesarias mejoras desde el punto de vista de diseño y eficiencia de la interfaz y estas mejoras han de llevar un tiempo que alargaría de forma innecesaria las conclusiones que en el estado actual de la investigación pueden extraerse al respecto de la tecnología desarrollada. Puede llegar a decirse que el desarrollo de una interfaz que incorpore mejoras como las que planteamos en las líneas de trabajo futuras del capítulo siguiente, junto con la evaluación de la herramienta en situaciones reales de docencia, con grupos experimentales y de control que proporcionen resultados estadísticos significativos sobre la misma, podría constituir en sí misma el trabajo de investigación de una segunda tesis doctoral.

En definitiva, las pruebas que hemos diseñado y que nos ha parecido razonable realizar desde el punto de vista de evaluación de la tecnología han consistido en el desarrollo de un **Curso Básico de Ejercicios de Cálculo Integral**. A lo largo del presente capítulo vamos a ir exponiendo cuestiones relevantes que han ido surgiendo durante las pruebas. Algunas de ellas son francamente positivas. Sin embargo, y decir lo contrario sería faltar a la verdad, hay cuestiones en las que hemos comprobado que aun es posible incorporar nuevas mejoras a *MathEdu* que pueden hacer de la herramienta un sistema más útil y eficiente.

4.1. Contenido del curso.

Dentro de la enorme cantidad de ejercicios de integración que se pueden encontrar en textos específicos sobre el tema como pueden ser [R.A.E.C., 1971], [Bronte, 1974] o [Diego, 1987], creímos conveniente centrarnos en un grupo significativo y que no hiciese interminable el proceso de diseño de ejercicios. Por este motivo escogimos como fuente de información para el desarrollo del curso el texto [Spivak, 1980], en concreto su capítulo 18 sobre *Integración en términos elementales*. Con el desarrollo de un curso que abarca todo un capítulo de un libro prestigioso como es el *Calculus* de Spivak pensamos que en cierto modo estamos tratando aspectos que están más próximos a la didáctica que a la tecnología. Es más, el curso cubre una buena parte del contenido de una aplicación que actualmente está comercializada y que ya apareció en otras partes de la memoria, *Calculus Wiz*. Pensamos que el desarrollo de un conjunto de pruebas que abarcan, si no la totalidad, si una parte importante de una aplicación profesional en uso, sirve como garantía de la validez de la tecnología desarrollada.

Volviendo al objeto del presente apartado, en este capítulo se describen los siguientes métodos de resolución de integrales indefinidas:

a) Integración por partes,

b) Integración por sustitución (cambios de variable),

c) Integración de funciones trigonométricas,

d) Integración de funciones racionales.

A esta lista de integrales añadimos propiedades de simplificación de integrales así como un conjunto de integrales inmediatas. Por tanto el aspecto que presenta la paleta de estrategias del curso que hemos diseñado como pruebas del sistema es la misma que ha ido apareciendo a lo largo de la memoria y que recordamos en la ilustración 4.1

Ilustración 4.1. Paleta de estrategias en *MathEdu*.

En lo que resta del capítulo vamos a exponer, una por una, cada una de las estrategias enumeradas. Comentaremos los aspectos más relevantes del diseño (dificultades, hallazgos relevantes, etc.) e indicaremos ejemplos del tipo de ejercicios que se pueden resolver.

4.2. Simplificación de integrales

Con objeto de ser capaces de resolver integrales que pueden simplificarse aplicando las propiedades de linealidad del operador integral (multiplicación por escalares y descomposición de la integral de una suma en suma de integrales), incluimos sendas estrategias para resolver estas propiedades. Denominamos las estrategias como *propiedad lineal* y *propiedad escalar*. Los patrones estructurales de estas estrategias son los siguientes. Para la estrategia *propiedad lineal*

$$\int Plus(f\_)\,dx$$

En el caso correspondiente a esta propiedad, la metavariable *f* se define de la siguiente forma

{*f*,
knownPlusQ,
funcionesAptasGen}

El predicado *knownPlusQ* analiza el integrando comprobando que está formado por la suma de cualesquiera funciones cuyas definiciones están incluidas dentro del tipo de ejercicios de integración y que, en consecuencia, son reconocibles mediante sus correspondientes predicados. Es decir, que si aparece una función que no es reconocible con ninguno de los predicados definidos en el resto de estrategias de integración, la propiedad no sería aplicable. En caso de que se pudiese aplicar la estrategia *propiedad lineal* el sistema se encontraría con un sumando correspondiente a una integral que no sabría resolver.

Como se puede apreciar en el patrón estructural, la metavariable *f* aparece afectada de dos caracteres de subrayado. El patrón condicionado resultante permite la aparición de dos o más instancias de funciones con las características de *f*, dado que a priori es imposible saber cómo son las expresiones que pueden resultar en la generación o en la resolución interactiva y por ello es necesario enunciar estructuras generales de reconocimiento.

En cuanto a la generación, se usa igualmente un generador que utiliza exclusivamente las funciones de generación declaradas en el tipo de ejercicio integración, al objeto de que los ejercicios que se planteen al alumno sean resolubles coherentemente.

Con respecto a las acciones de esta estrategia, sólo hemos declarado una acción de entrada de expresiones. Esta acción le pide al alumno que introduzca la integral simplificada descompuesta en todos los sumandos que sea posible obtener. El dato introducido por el alumno se verifica mediante comparación con la correspondiente expresión objetivo de la acción.

Con respecto a la segunda estrategia de simplificación, la *propiedad escalar*, se define el patrón estructural de la misma como

$$\int (a\_ \cdot f\_)dx$$

En el caso correspondiente a esta propiedad, la metavariable *a* se define de la siguiente forma

$\{a,\ ConstQ,\ integerGen(\{\ \})\}$

El predicado *ConstQ* permite la aparición dentro del integrando de cualquier símbolo o cualquier valor numérico del tipo que sea. Excluye explícitamente en su implementación la variable x al ser la variable de integración. Para la generación únicamente se incluyen integrales que lleven asociada una constante entera en el integrando.

En cuanto a la definición correspondiente a la metavariable *f* es semejante a la dada para la estrategia de linealidad anterior y su justificación es la misma.

$\{f,\ knownQ,\ funcionesAptasGen\}$

Como única acción se le pide al alumno, al igual que en caso anterior, que aporte al sistema la expresión simplificada resultante, verificándose la validez de la misma.

Existen otras simplificaciones que no hemos introducido en el sistema pero que sería necesario hacerlo a fin de dotar al curso de una mayor completitud. Estas simplificaciones corresponden a esquemas como producto de polinomios, funciones racionales reducibles, potencias de sumas, etc. Algunos ejemplos de estas situaciones son

$$\int (x+1)(x-1)\,dx;\ \int \frac{(x^2-1)}{(x-1)}dx;\ \int (1-Sen^2(x))^2\,dx$$

4.3. Integrales inmediatas.

Dentro de este grupo hemos incluido aquellas integrales que suelen considerarse habitualmente en los textos como integrales de resolución *inmediata* y que constituyen un grupo (generalmente no homogéneo) de integrales cuya solución se obtiene sin ningún tipo de manipulación o sustitución, de forma directa. En concreto las que hemos considerado en nuestro curso son las que aparecen en el capítulo 18 del *Calculus*:

a) $\int dx = x$

b) $\int x^n dx = \frac{x^{n+1}}{n+1}; \quad n \neq -1$

c) $\int \frac{dx}{x} = Ln(x)$

d) $\int e^x dx = e^x$

e) $\int Sen(x) dx = -Cos(x)$

f) $\int Cos(x) dx = Sen(x)$

g) $\int \frac{dx}{Cos^2(x)} = Tg(x)$

h) $\int \frac{dx}{1+x^2} = arcTg(x)$

i) $\int \frac{dx}{\sqrt{1-x^2}} = arcSen(x)$

El caso a) está levemente modificado dado que en el texto se considera la integral de una constante cualquiera. Dado que hemos incluido una estrategia de simplificación de integrales en las que en el integrando aparecen constantes (escalares o símbolos cualesquiera), el caso a) tal y como aparece propuesto en el texto se resuelve, en nuestro caso, aplicando la propiedad modelada. El modelado de estos ejercicios no resultó ser excesivamente complicado salvo algunos detalles significativos que vamos a comentar a continuación.

La primera cuestión relevante es acerca de la estrategia común que representa a todos ellos. Inicialmente realizamos un modelado en el que discriminamos por los patrones estructurales de cada uno de los ejercicios, llevados por la idea de que al ser ejercicios con integrandos con distintas estructuras y por tanto con distintas metavariables, debían resolverse de forma distinta. Pero la realidad es que la forma de resolver estos ejercicios es la misma para todos ellos y consiste, como integrales inmediatas que son, en introducir la solución como única acción relevante. Decidimos por tanto declarar sólo una estrategia que represente a dichas integrales. Al declarar una estrategia para integrales con patrones tan

dispares hay que resolver el problema de cómo declarar las metavariables, ya que el patrón estructural para todas ellas es único. En concreto es

$$\int u\_dx$$

y en la declaración de la metavariable *u* hay que incluir los distintos casos representados por las integrales enumeradas más arriba. Esto nos planteó una dificultad adicional ya que el papel de la metavariable *u* no era el que habíamos previsto al ser una composición de metavariables. Para aclarar mejor esta circunstancia vamos a detallar en términos de metavariables cómo sería la representación de una de las integrales anteriores, por ejemplo la segunda. Para generalizar una integral como por ejemplo

$$\int x^2 dx$$

es necesario disponer de una metavariable que represente la constante que aparece en el exponente. El modelo es pues

$$\int x^{n\_} dx$$

siendo

$$\{n, IntegerQ, integerGen(\{-1\})\}$$

A partir de este modelo, que fue el que planteamos inicialmente, hubo que obtener el siguiente, que agrupa en una única metavariable toda la información concerniente al integrando. El modelo queda

$$\int u\_\, dx$$

siendo la declaración de la única metavariable *u*

$$\{u,\ AlternativeQ(\{MonomialQ, RatQ\})\ x^{integerGen(\{-1\})}\}$$

En la expresión anterior es relevante que comentemos los aspectos relativos a la condición que caracteriza la metavariable. Se observa que la condición en este caso ya no es un predicado simple como en los diversos ejemplos que han ido surgiendo a lo largo de la memoria, sino que es un predicado compuesto que involucra tanto a una función

polinómica (en forma de monomio) como a una función racional (según el exponente *n*) de forma alternativa.

Una posible mejora que detectamos al realizar las pruebas consiste en desarrollar un mecanismo general que a partir de patrones correspondientes a distintos tipos de ejercicios permita al diseñador agruparlos en patrones compuestos/complejos que involucren a los patrones simples. Formalmente consistiría en hacer algo como lo siguiente. Sean los patrones y declaraciones de la tabla 4.1

Estrategias	Patrón estructural	Declaración de metavariables
E_1	$P_1(m_1, m_2)$	Caso 1: $(m_1, c_1^1, g_1^1); (m_2, c_2^1, g_2^1)$
		Caso 2: $(m_1, c_1^2, g_1^2); (m_2, c_2^2, g_2^2)$
E_2	$P_2(m)$	(m, c, g)

Tabla 4.1. Patrones previos a la composición.

En un momento dado el diseñador puede estar interesado en agrupar las estrategias anteriores en una estrategia única con un único patrón estructural, *P*, de tal forma que la estrategia correspondiente incluya a las dos iniciales como casos particulares. La tabla 4.2 resume este hecho.

Estrategia	Patrón estructural	Declaración de metavariables
E	$P(u\_)$	$\left(u, \left((c_1^1 \wedge c_2^1) \vee (c_1^2 \wedge c_2^2)\right)(\pi_{P_1}), P_1\left((g_1^1 \wedge g_2^1) \vee (g_1^2 \wedge g_2^2)\right)\right)$
		$\left(u, c\left(\pi_{P_2}\right), P_2(g)\right)$

Tabla 4.2. Resultado de la composición de patrones.

Vamos a explicar brevemente la tabla anterior. Se trata de describir la estrategia E, cuyo patrón estructural consta de una única metavariable, *u*, definida para cada uno de los casos que aglutina. Comenzamos describiendo el segundo caso por ser más simple. En dicho caso la metavariable *u* se genera aplicando el generador *g* dentro del patrón P_2. En cuanto a la condición, primero se debe verificar que la expresión a analizar cumple el patrón P_2 (lo denotamos como π_{P_2}) y entonces se verifica la condición *c* sobre la expresión que cumple dicho patrón. Veamos un ejemplo.

Supongamos que P_2 es $m\_ + m\_^2$ y que c es LogQ, de manera que la estrategia se aplicará cuando la expresión dependa de una función logarítmica y de su cuadrado. Para comprobar que la expresión $\ln x + (\ln x)^2$ cumple $c(\pi_{P_2})$ se verifica primero que

$$MatchQ\left(\ln x + (\ln x)^2, m\_ + m\_^2\right) = True$$

tras lo cual se toma $m = \ln x$ y, acto seguido,

$$LogQ(\ln x) = True$$

con lo que el patrón de la estrategia E se cumple.

El otro caso es más complejo aunque la idea es similar. La complejidad se debe a la circunstancia de que hay dos casos posibles y dos metavariables implicadas en cada caso. Con la notación $P_1\left((g_1^1 \wedge g_2^1) \vee (g_1^2 \wedge g_2^2)\right)$ para la generación de la metavariable u estamos indicando que o bien se usan los generadores g_1^1 y g_2^1 o bien los generadores g_1^2 y g_2^2 aplicados en cualquiera de los dos al patrón P_1. En cuanto al reconocimiento de estrategias, $\left((c_1^1 \wedge c_2^1) \vee (c_1^2 \wedge c_2^2)\right)(\pi_{P_1})$ indica que habiéndose verificado el patrón estructural P_1 para alguna expresión dada (π_{P_1}), se cumplen alguno de los pares de condiciones relativas al patrón P_1, y en tal caso se cumple la estrategia E.

Hay un hecho que queremos resaltar relacionado con aspectos didácticos más que con aspectos puramente tecnológicos. Se trata del hecho de cómo el sistema puede ayudar al profesor a detectar fallos en el diseño. La cuestión que nos sucedió fue que al modelar las integrales del apartado b) olvidamos incluir la condición de que la potencia n debía ser distinta de -1. Una vez que concluimos el modelo y cuando se estaba probando su funcionamiento con ejemplos generados aleatoriamente sucedió que por ejemplo una integral como

$$\int x^{-1} dx$$

tras aplicar un cambio de variable el sistema nos permitía resolverlas como un monomio en x (caso b) o como un logaritmo (caso c). La primera opción era incorrecta pero como el sistema detectaba el patrón estructural y verificaba el patrón condicional de dicho caso, no ponía objeción alguna a introducir como solución

$$\frac{x^0}{0}$$

Y, cómo no, al existir una división por cero *Mathematica* advertía de esta situación anómala. Una vez que revisamos las causas que provocaban esta situación y pusimos remedio, el sistema ya sólo aceptaba que la integral anterior se resolviese como un logaritmo, como realmente debe ser.

En este ejemplo acabamos de ver cómo en determinadas situaciones el propio sistema hace reflexionar al profesor sobre el modelado que hace de los ejercicios y los errores que se pueden producir durante el mismo, que a posteriori se traducen durante la fase de resolución interactiva, en comportamientos anómalos del sistema.

4.4. Integración por partes.

Dentro de esta estrategia nos hemos limitado a integrales cuyos integrandos poseen una estructura determinada formada por funciones en las que siempre aparece como factor dentro del integrando un polinomio y como un segundo factor aparece alguno de los siguientes tipos de funciones:

a) trigonométricas
b) exponenciales
c) logarítmicas

Hemos dejado sin tratar, puesto que conceptualmente no aportan nada nuevo, los casos correspondientes a las integrales

$$\int Sen(nx)Cos(mx)dx \text{ y } \int Sen(nx)e^{ax}dx$$

En el capítulo 3 se pueden ver detalles relativos al diseño de ejercicios que conciernen a esta estrategia. En el modelado de esta estrategia de resolución hemos considerado por consiguiente tres casos distintos para el patrón estructural de la fórmula involucrada en el mismo. Estos casos corresponden a cada uno de los tipos de funciones anteriores, con la salvedad de que en las funciones exponenciales hubo que distinguir entre las de base decimal y las de base natural. El patrón estructural es común a todos los casos y utiliza dos metavariables. La primera de ellas representa siempre el factor polinómico y la segunda el otro factor. Su declaración es

$$\int (u\_ \cdot v\_)dx$$

En consecuencia, las declaraciones de la metavariables para cada uno de los casos es la que sigue a continuación. La metavariable u se declara igual en los tres casos. Tiene por condición ser un polinomio y, como función generadora, la función generadora de polinomios de grado2 sin término independiente. Es decir

$$\{u,\ PolynomialQ, polyGen(2,0)\}$$

La metavariable v es la que va a diferenciar entre unos ejemplos y otros. Contempla las distintas funciones a que acabamos de hacer referencia más arriba. Se declara, en cada uno de los casos, como sigue:

a) Para el caso de funciones trigonométricas:
$$\{v, trigQ, simpleTrigGenerator(any)(x)\}$$

b) Para el caso de funciones exponenciales, de base natural:
$$\{v, ExpQ, e^{polyGen(1,0)}\}$$

c) Para el caso de funciones exponenciales, de base decimal:
$$\{v, ExpDecQ, Random(Integer, \{2,10\})^{polyGen(1,0)}\}$$

d) Para el caso de funciones logarítmicas:
$$\{v, LogQ, Log(polyGen(1,0))\}$$

Como podemos observar, la diferencia entre los casos b) y c) estriba en dos hechos. Por un lado en el condicionante de la metavariable, siendo dos predicados diferenciados para cada caso. Por otro lado las funciones generadoras también son distintas pues en el caso de funciones de base decimal, esta se genera de modo aleatorio con valores naturales entre 2 y 10.

Con respecto a la declaración de las metavariables que constituyen los patrones asociados a esta estrategia no es necesario añadir nada más.

Como ya vimos en el epígrafe 3.3, las estrategias se caracterizan por los patrones estructurales, los cuales se definen en función de las metavariables; y por los tipos de acciones de resolución, que determinan el modo en que debe aplicarse la estrategia para resolver el ejercicio. En la resolución de ejercicios por partes es necesario siempre resolver una segunda integral, generalmente más simple que la del enunciado, y para efectuar esta acción es necesario incluir, entre otras, la acción de *resolución de ejercicio*, entre los tipos de acciones que hemos declarado para resolver este tipo de ejercicios. A priori no es posible determinar qué estrategia será válida para este subproblema ya que es común que resulte ser una integral inmediata o bien, también es frecuente, tener que volver a aplicar la estrategia

por partes. Así pues la acción de resolución de ejercicio sólo establece que es necesario resolver un nuevo ejercicio de tipo integración y se pasan los datos oportunos en los parámetros de entrada y salida. El resultado de la declaración de esta acción se concreta en la estructura de datos que aparece a continuación. Consta de cuatro parámetros que representan: a) un mensaje al alumno sobre lo que debe realizar, b) el tipo de ejercicio que se va a invocar, c) los datos de entrada a dicho tipo de ejercicio y d) los datos de salida del mismo.

resolver ejercicio("Ahora debes resolver la integral de [Vdu]",
integración,
entrada(integral → $\int(V \cdot du)dx$*)*,
salida(resultadoIntegral → *solución)*)

Observemos que en tiempo de resolución los valores de las variables V y du definidas durante la resolución a partir de las metavariables u y v respectivamente, son los que se utilizan como datos de entrada al subproblema. La *solución* del mismo es el dato de salida que se utiliza como resultado de la integral resuelta. Este mecanismo nos permite encadenar la resolución de ejercicios basándonos en ejercicios de otros (o del mismo) tipos.

4.5. Integración de funciones racionales.

Hemos alterado el orden de estrategias de resolución que aparece al inicio del capítulo porque para la resolución de integrales trigonométricas necesitábamos poder resolver integrales racionales. Es frecuente que al efectuar un cambio de variable o por sustitución en una integral trigonométrica, la integral resultante resulte ser una integral racional y, en consecuencia, era necesario disponer de este método de resolución antes de modelar otras estrategias.

El modelo de datos para este tipo de ejercicios es el siguiente. En primer lugar describimos cómo es el patrón estructural y, posteriormente la declaración de metavariables. El patrón que representa el cociente de dos polinomios es

$$\int(\frac{p\_}{q\_})dx$$

La declaración de las dos metavariables que aparecen en el integrando es la siguiente. La metavariable p que representa el polinomio del numerador se genera simplemente como un polinomio hasta grado 2 con término independiente

$$\{p, PolynomialQ, polyGen(2,1)\}$$

La metavariable *q* que representa el polinomio del denominador es de características similares.

$$\{q,\ PolynomialQ,\ polynomial(Random(Integer,\{Exponent(p)+1,10\}))\}$$

Un detalle a tener en cuenta está asociado a la función generadora de polinomios. Hasta el momento veníamos utilizando como función generadora *polyGen(grado_, término_)*. El problema que nos surgió con esta función fue que los polinomios que se generaban daban como raíces valores reales o complejos de cualquier índole lo que hace muy poco atractiva (desde el punto de vista didáctico) o incluso imposible su manipulación por parte del alumno. Por eso fue que tuvimos que afinar más y definir la función *polynomial(grado_)* en el núcleo de *MathEdu* en la que los polinomios que se generan tienen raíces reales y/o complejas con coeficientes enteros. El número de raíces reales, complejas o de ambas que se generan se determina también de forma aleatoria.

Una restricción que hemos impuesto es que en los ejercicios que se generan siempre es mayor el grado del denominador que el del denominador para evitar tener que realizar el cociente de polinomios. La mejora obvia de esta restricción se obtendría definiendo un nuevo tipo de ejercicio denominado, por ejemplo, *cocientePolinomial* y cuyo objetivo fuese expresamente servir de modelo a ejercicios relativos al cociente de polinomios cuando en el integrando el grado del numerador fuese mayor o igual que el del denominador. Como resultados de este tipo de ejercicios se obtendrían tanto la fracción racional resultante como el polinomio resto del cociente.

Un problema adicional que nos planteó el modelo para este tipo de integrales fue a la hora de preguntar al alumno por las raíces del polinomio denominador. Al ser la generación aleatoria el profesor desconoce a priori cuántas raíces y de qué tipo van a obtenerse. Pensamos cómo conseguir, con los tipos de acción de que disponemos, averiguar el número de raíces y preguntárselas al alumno, para ver si había realizado bien los cálculos e iba encaminado correctamente hacia la descomposición en fracciones simples. Como ya hemos visto en el capítulo 3 existe una acción de asignación de valores pero no tenemos diseñada ninguna acción condicional del sistema para plantear distintos cursos de acción. La solución llegó desde el propio *Mathematica*. Resolver ecuaciones es una tarea simple mediante la función **Solve**. La alternativa entonces fue

1. resolver la ecuación $p(x)=0$. *Mathematica* proporciona una lista con las raíces.
2. pedir al alumno mediante una acción de entrada de expresiones que introdujera dicha lista.

Una alternativa muy elegante a esta segunda acción habría sido, caso de haber tenido desarrollado un curso sobre resolución algebraica de ecuaciones, hacer una llamada de *resolución de ejercicio* mediante la que se propusiese resolver el subproblema correspondiente a descomposición en factores de un polinomio.

No previmos que si los valores estaban en distinto orden, aun siendo estos los mismos, el sistema no daba por buena la contestación del alumno. A partir de este momento comenzó a cobrar sentido el diseño de una acción nueva que permitiera al alumno teclear un conjunto de datos y que el sistema los validara independientemente de su ordenación. Surgió así la acción de *conjunto de datos* que se incorporó al conjunto de acciones que ya hemos explicado anteriormente. La estructura de esta acción es similar al de la acción de *entrada de expresiones* con la diferencia de que la expresión que admite es un conjunto de elementos separados por comas como puede verse en la ilustración 4.2

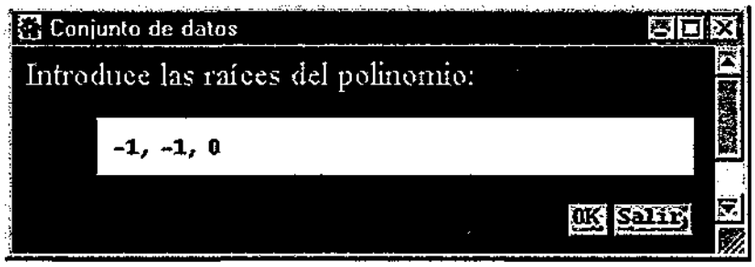

Ilustración 4.2. Entrada de datos.

El orden en que se introducen los datos es irrelevante, el sistema se encarga de valorar el conjunto y decidir si es o no correcto.

Para poder estudiar la descomposición en fracciones que el alumno tiene que hacer del integrando que se le ha presentado ha sido necesario utilizar acciones de entrada de patrones en las que las condiciones venían representadas por patrones estructurales. Este tipo de acciones nos obliga posteriormente a preguntar al alumno, utilizando una acción de conjunto de datos, por los coeficientes de las fracciones, para verificar que no sólo ha sido capaz de plantear desde el punto de vista de su estructura las fracciones simples, sino que también ha sabido calcular los coeficientes indeterminados asociados a las mismas. En este sentido la acción de entrada de patrones siguiente

"Introduce la expresión de la descomposición en fracciones simples del integrando para el caso de raíces reales simples".

está orientada a que el alumno teclee una expresión similar a

$$\frac{A}{x+7}+\frac{B}{x+10}$$

siendo irrelevante el orden de los sumandos y los símbolos con que denota los numeradores de ambas fracciones simples. El patrón estructural asociado que permite el reconocimiento es

Plus[_?*simpleQuotientQ*, _?*simpleQuotient* ..]

En esta expresión el predicado _?*simpleQuotientQ* comprueba que la expresión que teclea el alumno está formada por una suma de fracciones cuyos numeradores son símbolos cualesquiera y los denominadores son expresiones binómicas.

Al hilo de estas cuestiones el uso de las acciones de entrada de patrón nos plantea un nuevo problema no previsto a la hora de utilizar *MathTrainer*. La resolución guiada de ejercicios utiliza la información contenida en las acciones para generar los datos que el alumno debería calcular en una hipotética sesión de resolución interactiva. Pero *Mathematica* puede realizar cálculos de expresiones que tengan un significado matemático concreto. No puede efectuar cálculos de expresiones usando exclusivamente patrones estructurales. Por este motivo *MathTrainer* no es capaz de resolver las expresiones que van ligadas a este tipo de acciones. Necesita, por tanto, un método para calcular un valor asumible en la explicación de la resolución del ejercicio planteado. La solución a esta situación pasa por incorporar a la declaración de este tipo de acciones en la fase de diseño con *MathEdu Designer* un nuevo campo que permita especificar cómo obtener las expresiones asociadas. Sin embargo para el prototipo actual de la herramienta no hemos incluido dicha mejora.

Como es bien conocido, la resolución de integrales racionales requiere efectuar una descomposición en fracciones simples del integrando. Dicha descomposición depende del numero y tipo de raíces que aparecen en el polinomio del denominador. La complejidad de las expresiones resultantes también depende de este hecho. Existe un método alternativo, denominado método de *Hermite*, para situaciones en las que existen raíces con multiplicidad mayor que uno, especialmente si son complejas. Naturalmente plantear la posibilidad de obtener una descomposición en fracciones simples de un ejercicio generado aleatoriamente del que desconocemos número y tipo de raíces para poder calcular la integral asociada parece, a primera vista, una tarea casi imposible. Al igual que en el caso de la generación de polinomios, hemos desarrollado un conjunto de funciones que hemos incluido en el núcleo de *MathEdu*, con el fin de sea posible obtener la descomposición en situaciones generales.

Para concluir la exposición de esta estrátegia de resolución vamos a enumerar el conjunto de acciones que hemos modelado y que se formulan al alumno en tiempo de resolución interactiva, indicando para cada pregunta el tipo de acción a que corresponde y el valor de la respuesta correcta que se debe introducir. Para facilitar la exposición vamos a suponer un ejercicio típico de integración racional como puede ser

Calcular $\int \frac{2x-4}{x^2+17x+70}dx$

mensaje: Al ser la integral de la función racional $\frac{2x-4}{x^2+17x+70}$ vamos a efectuar una descomposición en fracciones simples del integrando.

datos: Calcula e introduce las raíces del polinomio en el denominador
valor: -7,-10

alternativa: ¿Las raíces tienen multiplicidad simple?. Selecciona una opción en la paleta.
valor: SI

patrón: Introduce la expresión de la descomposición en fracciones simples del integrando para el caso de raíces reales simples.
valor: $\frac{A}{x+7}+\frac{B}{x+10}$

patrón: Ahora debes efectuar la suma de las fracciones simples. Expresa el numerador de la expresión resultante como un polinomio en x.
valor: $\frac{(A+B)x+10A+7B}{(x+7)(x+10)}$

mensaje: Para calcular los coeficientes de las fracciones simples debes igualar los coeficientes de los términos de igual grado en los numeradores del integrando y de la fracción suma que acabas de obtener. Resuelve el sistema de ecuaciones resultante.

datos: Introduce los coeficientes indeterminados
valor: -6, 8

expresión: Introduce la suma de integrales resultante en función de las fracciones simples.
valor: $\int \frac{-6}{x+7}dx+\int \frac{8}{x+10}dx$

expresión: Introduce el resultado de $\int \frac{-6}{x+7}dx$
valor: $-6Log(x+7)$

expresión: Introduce el resultado de $\int \frac{8}{x+10}dx$
valor: $8Log(x+10)$

expresión: La solución es:

valor: $-6Log(x+7)+8Log(x+10)$

Es importante destacar que dado el carácter simbólico de los datos anteriores todas las expresiones admiten otras formas. Por ejemplo la solución final puede escribirse de estos dos modos alternativos

$$8Log(x+10)-6Log(x+7)$$

o bien

$$-6Log(7+x)+8Log(10+x)$$

e inclusive

$$Log\left(\frac{(x+10)^8}{(x+7)^6}\right)$$

MathEdu admite como válidas las correspondientes expresiones, y, por tanto, la equivalencia entre todas las soluciones.

4.6. Integración por sustitución.

La estrategia de integración por sustitución de variables es, con diferencia, la más problemática de las que nos hemos planteado. Son varios los aspectos que dificultan la modelización de este método. Vamos a ir explicando desde dos puntos de vista cada uno de los puntos que enumeramos a continuación. Por una parte explicaremos dónde surge la dificultad del modelo y, por otra, la solución que hemos aportado.

1. La generalización de enunciados
2. La generación de funciones susceptibles de ser integradas mediante este método
3. La identificación de condicionantes válidos para todas ellas

A menudo se considera la fórmula de sustitución como

$$\int f(g(x))\cdot g'(x)dx = \int f(u)du, \quad u = g(x) \qquad [1]$$

siendo el único requisito que f y g' sean funciones continuas. Con unas restricciones tan débiles, el conjunto de posibles funciones que verifiquen la expresión anterior es muy extenso. Esta circunstancia dificulta la declaración de un enunciado general válido para todas ellas, pues en principio cualquier composición que cumpla la expresión anterior vale y, en consecuencia, podemos encontrarnos con funciones tan dispares como pueden ser

logaritmos, polinomios, funciones trigonométricas exponenciales, irracionales, etc. Puesto que tras la sustitución la integral resultante debe ser una integral reconocible por el propio curso de integración para poder continuar con la resolución interactiva, hemos considerado ceñir la generación de la función f a los casos contemplados para la generación de integrandos en todas las estrategias modeladas. A la vista de estas consideraciones el modelo de patrón estructural viene dado por la expresión

$$\int f\_(g\_)\cdot gDeriv\_\, dx$$

que como podemos comprobar corresponde a la fórmula [1] anterior, siendo la definición de las metavariables la siguiente:

$$\{f, knownQ, funcionesAptasGen\} \qquad [2]$$

La definición de la metavariable g nos obliga a considerar funciones que sean derivables. Esta propiedad, aunque conceptualmente es clara, no es sencilla de estudiar. Requeriría implementar un paquete de funciones exclusivas para el estudio de la derivabilidad de una función y aún así sería excesivamente complejo adaptarlo a la generación de la metavariable g. Para solventar este problema y dado que los ejercicios que habitualmente se presentan a los alumnos en los textos involucran las funciones más comunes, trigonométricas, exponenciales, logarítmicas, irracionales, etc. decidimos implementar una sencilla función de generación que escoja aleatoriamente entre alguno de estos tipos de funciones. Por tanto g es

$$\begin{aligned}&\{g,\\ &AlternativeQ(\{TrigQ, LogQ, ExpQ, IrratQ, RatQ\}),\\ &RandomItem(\left\{Sen(x), Cos(x), Ln(x), e^x, \sqrt{x}, \frac{1}{x}, x^2+1\right\})\}\end{aligned} \qquad [3]$$

y *gDeriv*

$$\{gDeriv, EqualQ(D(g,x)), D(g,x)\} \qquad [4]$$

En la tabla 4.3 mostramos unos cuantos ejemplos que pueden resolverse con *MathEdu Solver* utilizando la estrategia de sustitución, los cuales se han obtenido a partir de las definiciones de las metavariables anteriores. Es sólo una pequeña muestra del conjunto de ejemplos diferentes que pueden llegar a plantearse de forma aleatoria y que el sistema sería capaz de interpretar como susceptibles de ser resueltos mediante esta estrategia. Si se desean más ejemplos sólo es necesario incrementar la lista de casos de g en [3].

$f(g\_)$	Ejemplos
$Sen(g\_)$	$\frac{Sen(Ln(x))}{x}$; $e^x \cdot Sen(e^x)$; $2x \cdot Sen(x^2+1)$
$Cos(g\_)$	$\frac{Cos(\sqrt{x})}{2\sqrt{x}}$; $-Sen(x) \cdot Cos(Cos(x))$
$e^{g\_}$	$\frac{e^{\sqrt{x}}}{2\sqrt{x}}$; $e^{Sen(x)} \cdot Cos(x)$
$Ln(g\_)$	$-Sen(x) \cdot Ln(Cos(x))$; $\frac{Ln(Ln(x))}{x}$; $\frac{-Ln(1/x)}{x^2}$
$\sqrt{g\_}$	$\frac{\sqrt{Ln(x)}}{x}$; $Cos(x) \cdot \sqrt{Sen(x)}$

Tabla 4.3. Ejemplos de funciones resolubles por sustitución.

Una observación que debemos hacer con respecto a la declaración de las metavariables en [3] es acerca de la función generadora *RandomItem*. La utilización de esta función combinada con la declaración del patrón estructural proporciona a *MathEdu Solver*, en el momento de la generación aleatoria de ejercicios, alguna de las funciones que aparecen descritas como ejemplos en la tabla 4.3. Se realizan por tanto diversas tareas de selección aleatoria de expresiones y de evaluación de patrones para obtener una expresión final que es la que se presenta al alumno en el cuaderno de resolución. De este modo logramos un buen número de casos de integrales que se resuelven mediante la estrategia de sustitución.

En cuanto a las acciones de resolución, segundo aspecto relevante en la declaración de estrategias, lo primero que debe hacerse es comprobar que el alumno sabe como efectuar la sustitución oportuna para obtener una integral simplificada. Para ello se le pide mediante acciones de tipo entrada de expresiones que especifique, en primer lugar, cuál es la sustitución que estima oportuno realizar (*t*) y la expresión derivada (*tDeriv*) de la sustitución que propone. En segundo lugar se le pide el resultado de la sustitución efectuada en la fórmula del enunciado inicial, el cual se almacena en la variable *integralSimplificada*. Una vez que el alumno ha introducido el cambio de variable que debe realizarse en las variables *t* y *tDeriv*, el sistema efectúa la sustitución de valores en la expresión de la integral inicial para generar la expresión de la nueva integral que se almacena en la variable indicada. La sustitución a que nos referimos se efectúa mediante la función de *Mathematica* **ReplaceAll** que presentamos en 2.5.2. Recordemos que esta función efectúa la sustitución de una expresión por otra dentro de una expresión simbólica general. Por tanto

$$\text{ReplaceAll}\left(\int e^{Sen(x)} \cdot Cos(x)dx,\ Rule(Sen(x),t)\right)$$

que se interpreta como "reemplazar en la expresión $\int e^{Sen(x)} \cdot Cos(x)dx$ todas las ocurrencias de $Sen(x)$ por t" produce la expresión

$$\int e^{t} \cdot Cos(x)dx$$

y si encadenamos a continuación reglas de sustitución similares a esta para la derivada de t (*tDeriv*) y para dx, obtenemos la expresión final que se asigna a la variable *nuevaIntegral*

$$nuevaIntegral = \int e^{t}\, dt$$

Al final del proceso ambas expresiones, *integralSimplificada* y *nuevaIntegral*, deben coincidir. Caso contrario el sistema advierte al alumno de que bien el cambio de variable propuesto o la simplificación efectuada han sido incorrectos. Por tanto una vez que se ha realizado la oportuna identificación del cambio de variable (la sustitución) que va a realizar el alumno y la ha realizado correctamente, se procede a invocar la resolución del nuevo problema simplificado. El nuevo ejercicio será de tipo integración, pero la estrategia que habrá que usar diferirá de la original. Generalmente se obtienen integrales más sencillas que las integrales iniciales.

Ejemplos	Sustitución	Nuevo ejercicio	Estrategia para el nuevo ejercicio
$e^{x} \cdot Sen(e^{x})$	$t = e^{x}$ $tDeriv = e^{x}$	$Sen(t)$	*Inmediata*
$\frac{Cos(\sqrt{x})}{2\sqrt{x}}$	$t = \sqrt{x}$ $tDeriv = \frac{1}{2\sqrt{x}}$	$Cos(t)$	*Inmediata*
$\frac{Ln(x)}{x}$	$t = Ln(x)$ $tDeriv = \frac{1}{x}$	$Ln(t)$	*Por partes*
$e^{Sen(x)} \cdot Cos(x)$	$t = Sen(x)$ $tDeriv = Cos(x)$	e^{t}	*Inmediata*
$\frac{Ln(Ln(x))}{x}$	$t = Ln(x)$ $tDeriv = \frac{1}{x}$	$Ln(t)$	*Por partes*

Tabla 4.4. Funciones resultantes tras el cambio de variable.

Podemos ver en la tabla 4.4 en qué nuevas funciones se han transformado algunos de los ejemplos de la tabla 4.3. Como la generación de los enunciados es aleatoria es imposible conocer a priori el tipo de integral resultante. Será el propio sistema el que se encargará de

validar las estrategias que el alumno proponga para la resolución de las integrales simplificadas.

Una vez que la integral inicial ha sido simplificada mediante el cambio de variable propuesto se procede a resolver el ejercicio resultante mediante el uso de la acción de *resolución de ejercicio*. La acción de resolución de ejercicio se declara como

resolver ejercicio("Ahora debes resolver la integral resultante de la sustitución",
integración,
entrada(integral → *integralSimplificada)*,
salida(resultadoIntegral → *solución)*)

En esta declaración hay que destacar el importante papel que juega la variable *integralSimplificada* que como hemos visto anteriormente representa el valor de la nueva integral que se debe resolver y que ha sido propuesta por el alumno y validada por el sistema. A partir de esta acción de resolución el sistema invoca un nuevo subproblema y el alumno deberá resolverlo indicando qué estrategia es la apropiada para el mismo y respondiendo a la secuencia de preguntas que se le planteen.

4.7. Integración de funciones trigonométricas.

Llegamos finalmente a las integrales de funciones trigonométricas propuestas en el libro de Spivak. Dentro de este tipo de integrales se consideran aquellas de la forma

$$\int Sen(x)^n Cos(x)^m dx$$

en las que pueden aparecer indistintamente el seno o el coseno como únicos factores o ambos simultáneamente. Además los valores de los exponentes n y m pueden ser arbitrarios. Dado que el método de resolución depende de la paridad de n y m, qué duda cabe que se complica enormemente el modelado de este tipo de integrales. Para ello hemos tenido que realizar un análisis detallado de las distintas posibilidades que existen y, para cada una de ellas, estudiar el tipo de resolución que debe plantearse.

La resolución de estas integrales se basa, esencialmente, en sustituciones y en simplificaciones que dan lugar a integrales inmediatas (definidas en el apartado 4.1.1 modelos e) y f)) o a integrales más sencillas que las originales. La forma de proceder, en líneas generales, consiste en analizar los exponentes del integrando y, dependiendo de su paridad, efectuar alguna sustitución determinada o realizar una simplificación previa para después efectuar la sustitución. Las simplificaciones y sustituciones a que nos referimos son

$$Sen^2(x) = \frac{1-Cos(2x)}{2},$$

$$Cos^2(x) = \frac{1+Cos(2x)}{2},$$

$$Sen^n(x) = Sen(x)\cdot Sen^{2k}(x) = Sen(x)\cdot(1-Cos^2(x))^k, \text{ (si } n \text{ impar)}$$

$$Cos^m(x) = Cos(x)\cdot Cos^{2k}(x) = Cos(x)\cdot(1-Sen^2(x))^k, \text{ (si } m \text{ impar).}$$

El modelo que hemos planteado para esta estrategia contempla todos los casos posibles, distinguiendo si las potencias que afectan a las funciones trigonométricas son pares o impares, ya que esa es la cuestión relevante de cara a la resolución. Su patrón estructural es

$$\int f\_dx$$

siendo la declaración de la metavariable *f*

$$\{f, ComposedTrigQ, ComposedTrigGenerator\}$$

El predicado *ComposedTrigQ* evalúa el patrón condicionado de la expresión que se muestra al alumno en el momento de la resolución y establece el caso a que corresponde (tabla 4.5).

Condición	patrones condicionados asociados
ComposedTrigQ	$Sen^{-?EvenQ}(x)$
	$Cos^{-?EvenQ}(x)$
	$Sen^{-?OddQ}(x)$
	$Cos^{-?OddQ}(x)$
	$Sen^{-?EvenQ}(x)\cdot Cos^{-?OddQ}(x)$
	$Sen^{-?OddQ}(x)\cdot Cos^{-?EvenQ}(x)$
	$Sen^{-?EvenQ}(x)\cdot Cos^{-?EvenQ}(x)$
	$Sen^{-?OddQ}(x)\cdot Cos^{-?OddQ}(x)$

Tabla 4.5. Declaración de la metavariable *f* como función trigonométrica.

Por lo que respecta a las acciones asociadas a esta estrategia, ya hemos comentado anteriormente cómo dependiendo de que la potencia del integrando sea par o impar (y según se trate de un seno o un coseno o un producto de ambos) los desarrollos que hay que efectuar en cada caso difieren del resto aun siendo similares entre todos ellos. Para conseguir controlar que en cada caso el alumno ejecute la simplificación adecuada el sistema debe detectar en qué caso se encuentra, analizando la información del integrando.

La identificación del caso apropiado se realiza mediante una acción de asignación de la forma

asignar(simplificación, determinarCaso(integral))

La función de determinación del caso establece, mediante comparación de patrones, el caso a que corresponde la integral propuesta y asigna a la variable *simplificación* el valor adecuado, en forma de regla de sustitución, de entre los que se muestran algunos ejemplos en la tabla 4.6 a continuación

Patrones	Reglas
$\int Sen^{\_?EvenQ}(x)dx$	$Sen^2(x) \to \frac{1-Cos2x}{2}$
$\int Cos^{m\_?OddQ}(x)dx$	$Cos^m(x) \to Cos(x)\cdot(1-Sen^2(x))^{\frac{m-1}{2}}$
$\int Sen^{n\_?OddQ}(x)\cdot Cos^{\_?EvenQ}(x)dx$	$Sen^n(x) \to Sen(x)\cdot(1-Cos^2(x))^{\frac{n-1}{2}}$
$\int Sen^{n\_?OddQ}(x)\cdot Cos^{m\_?OddQ}(x)dx$	$Sen^n(x) \to Sen(x)\cdot(1-Cos^2(x))^{\frac{n-1}{2}}$ o indistintamente $Cos^m(x) \to Cos(x)\cdot(1-Sen^2(x))^{\frac{m-1}{2}}$
$\int Sen^{\_?EvenQ}(x)\cdot Cos^{\_?EvenQ}(x)dx$	$Sen^2(x) \to \frac{1-Cos2x}{2}$ y $Cos^2(x) \to \frac{1+Cos2x}{2}$

Tabla 4.6. Un conjunto representativo de reglas de simplificación.

Una vez que se ha determinado la simplificación que debe realizarse se le pregunta al alumno, con la correspondiente acción de entrada de expresiones, por la simplificación que debe efectuarse, siendo la expresión objetivo la contenida en la variable *simplificación* definida en el paso anterior.

El siguiente paso es efectuar dicha simplificación. Antes de preguntar al alumno por la misma es necesario realizar los cálculos para saber a qué resultado debe llegar este. Mediante otra acción de asignación el sistema almacena en la variable correspondiente el resultado de la simplificación.

asignar(integralSimplificada, simplificar(integral, simplificación))

La función *simplificar* utiliza como parámetros el valor de *integral* y la variable *simplificación* anterior en la que está contenida la simplificación que debe realizarse. Recordemos que *integral* representa el símbolo en el que se guarda la fórmula del enunciado. La expresión resultante de este proceso se almacena en la variable *integralSimplificada*. En la tabla 4.7 esquematizamos el resultado de la simplificación/sustitución.

Valores de *n* y *m*		Estrategias para la integral simplificada resultante
n	*m*	
par	0	Inmediata y trigonométrica (iterando según grado de *n*)
0	par	Inmediata y trigonométrica (iterando según grado de *n*)
impar	0	Inmediata, Sustitución
0	impar	Inmediata, Sustitución
Par	Impar	Sustitución
Impar	Par	Sustitución
Par	Par	Sustitución, Trigonométrica
Impar	Impar	Inmediata, Sustitución

Tabla 4.7. Ejemplos de simplificación de integrales trigonométricas.

Una vez que el alumno ha introducido correctamente la integral simplificada (contenido en la variable *integralSimplificada*), es posible continuar con la resolución de la integral planteada inicialmente. Para ello, dado que la integral simplificada resultante va a tener que ser simplificada mediante las propiedades de linealidad de la integral introducidas al inicio del presente capitulo, planteamos una resolución de un nuevo ejercicio de integración en los términos siguientes

resolver ejercicio("Calcula la integral [*integralSimplificada*] que acabas de introducir",
integración,
entrada(integral → *integralSimplificada)*,
salida(resultadoIntegral → *solución)*
)

La integral simplificada habitualmente está compuesta de varios sumandos y tiene algún factor que afecta a todo el integrando. Por tanto el subproblema que se le va a plantear va a requerir que la estrategia de resolución inicial para el mismo sea alguna de las dos de

simplificación. Dependerá de los datos de cada ejercicio para que sea conveniente aplicar una u otra estrategias de simplificación.

Una solución alternativa consiste en pedir al alumno que, a partir de la integral simplificada anterior, realice los cálculos oportunos (los cuales ya son notablemente más sencillos) e introduzca el resultado final, el cual es fácilmente verificable mediante una acción de entrada de expresiones. Esta alternativa, cómoda desde el punto de vista del trabajo dedicado al modelado, es significativamente menos elegante. Por el contrario agiliza el modelado si lo que desea el profesor es comprobar cómo realizan los alumnos las simplificaciones ante los distintos tipos de integrales que se les planteen.

Al hilo de esta alternativa nos ha surgido una nueva idea que hasta el presente no habíamos considerado. Al igual que con *MathTrainer* es posible que sea el propio alumno el que proponga un ejercicio al sistema, esa posibilidad no la habíamos contemplado nunca para *MathEdu Solver*. Sería pues deseable poder plantear a *Solver* un ejercicio y que este lo reconociese e iniciara la resolución interactiva. Sería conveniente disponer de diversos controles en el cuaderno de resolución que permitan detener/continuar la resolución de un ejercicio (o avanzar/retroceder un paso, etc.) hasta que el alumno obtenga los resultados parciales de una cantidad indeterminada de ejercicios aún por resolver. Además, al ser diferentes las dificultades de los ejercicios que restan por resolver, es el propio alumno el que puede escoger entre introducir el resultado directamente (por ejemplo en el caso de las integrales inmediatas más simples), o bien hacer una resolución interactiva completa de un ejercicio. Los resultados parciales se podrían ir anotando sobre el cuaderno de resolución de un modo similar a una hoja de cálculo con celdas interactivas y el resultado final se podría obtener al combinar los resultados parciales de varias de las celdas interactivas. En este sentido, una parte del proyecto ENCITEC (subvencionado por la CICyT, TEL1999-0181) involucra a varios investigadores del grupo GHIA (Grupo de Herramientas Interactivas y Aplicaciones) que desarrollan una aplicación que permita relacionar información simbólica en distintas partes de un documento, el cual además se puede manipular a través de Internet.

Capítulo 5.

CONCLUSIONES

En este último capítulo vamos a presentar las conclusiones sobre nuestro trabajo. En primer lugar enumeraremos las principales contribuciones al *estado del arte* de los sistemas tutores inteligentes para la enseñanza de Matemáticas. En segundo lugar presentaremos una evaluación de los resultados alcanzados; y, finalmente, enumeraremos diversas mejoras que se pueden incorporar al sistema así como algunos trabajos futuros que tenemos planteado realizar.

5.1. MathEdu: un sistema para la enseñanza de Matemáticas.

A pesar de los grandes avances informáticos que paralelamente al cambio de siglo se han producido, es notorio el déficit de aplicaciones que permitan un grado suficiente de interactividad en el diálogo con el alumno en la docencia de Matemáticas. En la presente memoria hemos descrito una herramienta de autor, *MathEdu* basada en la *programación mediante ejemplos* y *Mathematica*. *MathEdu* permite construir, de manera relativamente simple y sistemática, colecciones de problemas de Matemáticas referentes a materias diversas que involucren cálculo simbólico; y la posterior resolución interactiva con el alumno de los ejercicios generados a partir de los ejemplos definidos por el profesor.

Es difícil concebir la resolución de problemas con un cierto grado de complejidad sin considerar, entre otras cuestiones, la posibilidad de especular sobre la existencia de distintas estrategias de resolución (Pitrat, 1990). *MathEdu* ha servido para aproximar los modos de razonamiento sobre la resolución de ejercicios de cálculo a la representación del conocimiento sobre los propios ejercicios. Fruto de dicha aproximación ha surgido, como resultado, un modelo de representación del conocimiento matemático extremadamente simple, en el que la idea de considerar las distintas estrategias de resolución es clave para la concepción global de los objetos representados.

En la representación y uso del conocimiento en *MathEdu* existen dos aspectos fundamentales:

- La capacidad de organizar la información genérica relativa a un tipo de ejercicio de cálculo, el objeto formal, y generar distintas instancias del mismo. Con *MathEdu Designer* hemos creado una herramienta de autor que permite representar elementos

matemáticos complejos que describan un tipo de ejercicio de cálculo y que sirven, a su vez, de modelo para generar instancias de los mismos.

- La capacidad de resolver dicho ejercicio mediante distintas estrategias estableciendo un diálogo con el alumno. Con *MathEdu Solver* se pueden resolver de forma interactiva los ejercicios que se plantean. *Solver* hace uso del *pattern-matching* y analiza la semántica de expresiones simbólicas para aceptar o rechazar la utilización de la estrategia propuesta por el alumno.

En consecuencia, una de las principales aportaciones del trabajo es la representación del conocimiento procedural necesario para la resolución de ejercicios. El motor de resolución es capaz de permitir al alumno utilizar indistintamente las posibles estrategias de resolución aptas para el ejercicio que se le propone. Esto se traduce en que, dependiendo de la estrategia escogida para resolver un ejercicio, el sistema plantea diferentes diálogos, de un modo absolutamente transparente para el alumno.

En resumen, las aportaciones fundamentales del trabajo que se acaba de presentar las podemos ubicar en dos planos diferentes:

Plano de representación del conocimiento

i. Creación de un modelo de representación de ejercicios de Matemáticas que involucran cálculo simbólico como objetos matemáticos complejos.

ii. Creación de un mecanismo de resolución de ejercicios generados aleatoriamente mediante acciones genéricas a los mismos.

Plano de interacción Hombre-Máquina

i. Creación de una herramienta de autor para crear representaciones abstractas de ejercicios que involucran cálculo simbólico (*MathEdu Designer*). Esta herramienta se basa en el paradigma de programación mediante ejemplos (Cypher, 1993).

ii. Creación de un motor de inferencia, *MathEdu Solver*, que utiliza las estructuras creadas por el profesor con *MathEdu Designer* para establecer un diálogo con el alumno que le guíe en la resolución del ejercicio aleatorio propuesto.

MathEdu integra con estos dos planos un conjunto de procedimientos y métodos que unifican en una sola herramienta la resolución interactiva de ejercicios. Dicha resolución involucra, por parte del alumno, la realización de cálculos simbólicos a través de un diálogo altamente elaborado. En definitiva con *MathEdu* conseguimos establecer una interacción

hombre-máquina para la resolución de ejercicios de Matemáticas que hasta la fecha no había sido realizada por ningún otro sistema en este contexto. En otras disciplinas sí que es posible encontrar aplicaciones con un mayor grado de interactividad que el que nosotros aportamos pero la auténtica novedad que aporta esta Tesis está en el tratamiento simbólico de objetos matemáticos complejos para conseguir la interacción entre el sistema y los usuarios (profesores o alumnos). Además la aportación del modelo de representación desarrollado pensamos que se puede adaptar a otras disciplinas científicas (Física, Química, etc.) para posibilitar la creación de cursos en las mismas.

5.2. Evaluación de resultados.

Con respecto a la evaluación de resultados la primera consecuencia relevante que extraemos es que es posible realizar colecciones de problemas de capítulos amplios de Matemáticas, tal y como acabamos de comprobar en el capítulo anterior. El prototipo no está desarrollado desde un punto de vista próximo a la Didáctica de las Matemáticas dado que no era el objetivo de la Tesis. Sin embargo satisface las necesidades del profesor en cuanto al modelado de ejercicios, lo cual consideramos que es un paso previo esencial de cara a acometer cualquier otro tipo de tratamiento de la información.

En cuanto al proceso de diseño hemos apreciado cierta complejidad inherente a un curso de integración. Antes de acometer el desarrollo del curso habíamos efectuado pruebas parciales, generalmente rápidas y simples, sobre el modelado de integrales inmediatas o por partes. Cuando nos hemos implicado en integrales más complejas hemos comprobado que es fundamental tener práctica en la resolución de ejercicios, tener claros los objetivos que se persiguen al resolver un ejercicio, qué formas diferentes (estrategias) hay para resolver un ejercicio, etc. Y, además de estas destrezas propias del profesor que desea realizar un curso, es importante tener un conocimiento suficientemente amplio de *Mathematica* ya que, como hemos podido ver en las últimas estrategias, es necesaria una labor de programación por parte del diseñador más importante de lo que inicialmente habíamos previsto. Otra alternativa posible puede ser dar apoyo de programación, a través de un trabajo en equipo, a la labor docente de diseño que debe realizar el profesor.

Desde el punto de vista del prototipo hemos percibido que la adaptación del mismo al diseño del curso ha sido buena. Sin embargo hemos detectado algunas carencias que vamos a resaltar a continuación. Las mismas son consecuencia de que *MathEdu* es un prototipo para el diseño y resolución interactiva de ejercicios. Todas ellas son subsanables y se incluirán las correspondientes mejoras en futuras versiones de la herramienta.

1. Sería útil contar con una acción de *iteración*, similar a las descritas en la sección 3.3, que permita hacer una misma operación un número indeterminado de veces. Como parte del desarrollo del prototipo hemos tenido que implementar

la acción *conjunto de datos* para poder introducir en el sistema una lista de datos que solvente la carencia de una acción de iteración como la que proponemos.

2. Como hemos comprobado en el capítulo 4 también han sido necesarios nuevos predicados que permitan obtener condiciones compuestas para funciones complejas. Ha quedado claro que los predicados básicos para polinomios, funciones trigonométricas, exponenciales o logarítmicas no eran suficientes.

3. No habíamos previsto la circunstancia que comentamos en el capítulo 4 respecto de la generación de expresiones en *MathTrainer* a partir de acciones de entrada de patrones. Hemos comprobado al valorar la estrategia de resolución de integrales racionales los problemas que surgen al tratar de efectuar una resolución guiada, pues no es posible generar expresiones simbólicas a partir de patrones estructurales. Este problema obligará a replantear dichas acciones incorporándoles información sobre la generación de expresiones válidas desde el punto de vista del patrón estructural que representan.

4. También hemos valorado la conveniencia de implementar un mecanismo de control del ejercicio (retroceder/avanzar un paso, etc.), así como de resolución de ejercicios propuestos por el alumno (en vez de generados por el sistema, del mismo modo que se hace en *MathTrainer*). Sin duda que este es un problema que se debe resolver en un futuro inmediato pues mejorará la eficacia y posibilidades de interacción alumno/sistema.

5.3. Trabajo futuro.

A lo largo del periodo de investigación seguido para la elaboración de esta memoria, hemos ido viendo distintos requerimientos y funcionalidades que han ido quedándose en el camino por diversos motivos. Qué duda cabe que futuras versiones de la herramienta deben incluir mejoras en todos los aspectos tratados en la misma, en ambos planos mencionados, de representación y de interacción. *MathEdu* actualmente forma parte del proyecto de investigación ENCITEC financiado por la CICyT (TEL1999-0181) que persigue la integración de distintas herramientas informáticas para la educación científico-técnica. Las mejoras y trabajos pendientes que consideramos más importantes van en la dirección que de un modo u otro han ido surgiendo dentro del proyecto y corresponden a las siguientes:

I. Diseñar una nueva interfaz que, sin perder las facilidades de manipulación de símbolos matemáticos que proporciona *Mathematica* actualmente, aporte una serie de facilidades de edición que mejoren el uso de la información. Una *interfaz* más

elaborada permitirá una comunicación con el alumno mucho más natural y cercana a la realidad del diálogo alumno-profesor.

En consonancia con el objetivo expuesto queremos volver a hacer referencia a *webMathematica* (accesible en http://www.wolfram.com/products/webmathematica/) presentado por Wolfram Research Inc., fabricante de *Mathematica*, en marzo de 2001. Este producto permite insertar comandos de *Mathematica* en páginas HTML de modo que, cada vez que la página es invocada, el código contenido en ella se ejecuta en un servidor de *Mathematica*. La gran aportación es que se puede ejecutar en cualquier *browser* al ser código HTML estándar. Desde la página anteriormente referida se puede acceder a ejemplos relativos a distintas disciplinas científicas. Desde el punto de vista de *MathEdu*, *webMathematica* tiene carencias notables: no existe ningún diálogo interactivo entre el sistema y el usuario ni se utilizan los mecanismos de metavariables, patrones, etc. que usamos en *MathEdu*. Pero lo realmente relevante desde nuestra perspectiva es ver cómo se pone de manifiesto desde el propio fabricante de *Mathematica* el interés en aportar herramientas de aprendizaje interactivo y preferiblemente si están accesibles a través de Internet. Aunque su funcionalidad sea muy reducida sin duda este producto viene a confirmar que el trabajo que estamos desarrollando en *MathEdu* va por el buen camino y que por tanto una de las futuras (e inmediatas) líneas de investigación debe ser la integración de la tecnología que hemos desarrollado para permitir su uso a través de Internet en una arquitectura cliente/servidor. De hecho una de las tareas incluidas dentro del proyecto ENCITEC tiene que ver con la gestión de expresiones simbólicas (fórmulas) autorreferenciadas dentro de un cuaderno a modo de hoja de cálculo inteligente, accesible a través de Internet. La incorporación a ese entorno de la funcionalidad de *MathEdu* es otro objetivo a abordar a corto plazo.

II. Incluir un Modelo del Alumno (Self, 1974), (Carro, 1999), que determinará las capacidades y destrezas alcanzadas por cada uno de los usuarios de la aplicación, a fin de presentarle nuevos ejercicios adecuados a las capacidades demostradas. De esta forma *MathEdu* puede transmitir confianza al alumno al corregirle los errores y adaptarse progresivamente a sus conocimientos. Con ello el alumno verá la aplicación no como un mero programa de cálculo simbólico, sino como un colaborador de su aprendizaje que le motivará a su utilización.

Un aspecto mucho más técnico que los que acabamos de indicar hace referencia a los procesos de *pattern-matching* que tienen lugar a la hora de interpretar la información subyacente en un ejercicio generado de forma aleatoria. Qué duda cabe que si bien lo logrado en esta primera versión aporta una nueva perspectiva en la resolución interactiva de ejercicios, no es menos cierto que al hacer uso continuado de la herramienta se nota una cierta rigidez en las formas y en la manipulación de datos. Por tal motivo un objetivo a plantear para lograr esto debe ser el siguiente:

III. Mejorar la interpretación de los patrones subyacentes en un enunciado. De este modo podrían enmascararse los enunciados utilizando propiedades matemáticas como la asociatividad o la conmutatividad de operaciones. *Mathematica* (y por tanto *MathEdu*) dispone de un mecanismo de aplicación de estas dos importantes propiedades. Sin embargo, cuando en la fase de diseño se representa una expresión, el modo en que se ha representado determina en el futuro cualquier ejemplo de dicha expresión, sin ser posible usar, en muchos casos, distintos modos de representación de un mismo objeto, cuestión esta que puede dar lugar a una mayor versatilidad en los enunciados y en los diálogos.

Como ejemplo que ilustre este último comentario podemos plantear el siguiente. Actualmente la generación de integrales resolubles por sustitución puede dar lugar a ejercicios como por ejemplo

Ejercicio. Calcular

$$\int \frac{dx}{x(1+\ln x)}$$

Este mismo ejercicio se puede plantear al alumno como

Ejercicio. Calcular

$$\int \frac{dx}{x+x\ln x}$$

en el que un simple cambio en el integrando hace que no sea tan evidente la presencia de una función y su derivada y obliga al alumno a un proceso de análisis de la información que se le suministra más complejo que en el primer caso.

Al hilo de lo anterior surgen ideas para dos nuevas vías de investigación más próximas a la Inteligencia Artificial que pueden nacer a partir de aquí. Por un lado se trataría de profundizar en la interpretación del enunciado de un ejercicio, posibilitando la creación de cursos sobre campos que no requieran estrictamente el uso de cálculo simbólico, por ejemplo en demostración de propiedades o teoremas, en el uso de gráficos; o bien de otras materias distintas de las Matemáticas (Bazin, 1993; Castells, 1993; Castells, 1993b, Mei-Chuen, 1999).

En segundo lugar consideramos que es importante indagar sobre la estructura de objetos matemáticos complejos, sus modos de representación y sus interacciones (Díez, 1996; Díez 1997). *MathEdu* puede servir de punto de arranque para este nuevo tipo de tareas en las que es clave disponer de una herramienta capaz de manipular descripciones de objetos matemáticos y dialogar con el usuario.

IV. Finalmente, será imprescindible más adelante, cuando concluyan algunas de las tareas enunciadas (fundamentalmente I y II), realizar unas pruebas de campo con profesores que diseñen un curso y alumnos que lo sigan y, entre todos, evalúen la eficacia de *MathEdu*.

APÉNDICES

APÉNDICE A

A) ESTRUCTURA DE DATOS DE LOS TIPOS DE EJERCICIO

En el apartado 3.3.2 de la memoria se hace referencia a la representación abstracta de la estructura de datos de un tipo de ejercicio. Dicha representación abstracta es la que se muestra en la figura A.1 a continuación. En negrita aparecen remarcadas las palabras clave de la estructura de datos.

```
id_tipo_de_ejercicio[
        enunciado,
        id_estrategia_1[
                etiqueta,
                descripción,
                formulaePatterns[Rule[id_formula, pattern]__],
                cases[{metaVarsSpec[__metaVariables]__}],
                resolutionActions[{
                        accion1,
                        accion2,
                        ... ,
                        accionN}]
        ],

        id_estrategia_2[... ],

        ... ,

        id_estrategia_n[... ]
]
```

Figura A.1. Estructura de datos de un tipo de ejercicio.

Esta estructura está formada por

1. el identificador del tipo de ejercicio
2. el enunciado correspondiente a dicho tipo
3. las estrategias válidas para ejercicios del tipo descrito, cada una de las cuales contiene, a su vez, la siguiente información
    a. una etiqueta descriptiva de la misma
    b. una descripción textual del uso que puede hacerse de la estrategia
    c. los patrones (estructuras formales abstractas) de las fórmulas del enunciado
    d. el conjunto de casos definidos para la estrategia, en función de las *metavariables* que se definan.
    e. Las acciones de resolución asociadas a cada estrategia

B) GENERALIZACIÓN DE CASOS DE EJERCICIOS PREDEFINIDOS

En el apartado 3.3.3.2 se hace referencia a la siguiente estructura, la cual muestra el contenido del tipo de ejercicio *integración* después de haber añadido un nuevo caso (resaltado en negrita). Con el nuevo caso introducido, los integrandos de los ejercicios que se generen constarán de un polinomio de grado 2 *sin* término independiente multiplicado por la función exponencial cuyo exponente es un polinomio de grado 1 sin término independiente.

```
Integración[
        "calcular $integral$",
        porPartes[
                "Por partes",
                "La integral se puede simplificar en una expresión más simple en función de
                factores derivables",
                formulaePatterns[integral → Integrate[Times[u_, v_], x],
                cases[{metaVarsSpec[
                        {{u, PolynomialQ, polyGen[2, 0]},
                        {v, TrigQ, simpleTrigGenerator[any][x]}},
                        {{u, PolynomialQ, polyGen[3, 0]},
                        {v, ExpQ, e^polyGen[1,0] }}
                }],
                resolutionActions[{
                        accion1,
                        accion2,
                        ... ,
                        accionN}
                        ]
]
```

Figura A.2. Ejemplo de estructura de datos para un nuevo caso de metavariables.

C) DEFINICIÓN DE NUEVAS ESTRATEGIAS DE UN TIPO PREDEFINIDO

La figura A.3 muestra la estructura de datos resultante tras la definición de una nueva estrategia (*racional*) para un tipo de ejercicio predefinido, tal y como se describe en el apartado 3.3.3.2.1 de la memoria. Hemos destacado en negrita el identificador del patrón de la fórmula con objeto de resaltar el hecho de que el enunciado, para un mismo tipo de ejercicio, sigue siendo el mismo, variando únicamente los datos relativos a las metavariables.

```
integración[
        "calcular $integral$",

        porPartes[
                "Por partes",
                "El integrando es de la forma f*g', siendo la derivada de f más sencilla que f y g'
                continua",
                formulaePatterns[integral → Integrate[Times[u_ , v_], x],
                cases[{
                        {u, PolynomialQ, polyGen[2, 0]},
                        {v, TrigQ, simpleTrigGenerator[any][x]}
                }],
                resolutionActions[{
                        accion1,
                        accion2,
                        ... ,
                        accionN}
                        ],

        racional[
                "Racional",
                "Integral de un cociente de polinomios",
                formulaePatterns[integral → Integrate[Times[P_ , Power[Q_ ,-1]], x],
                cases[{
                        {P, PolynomialQ, polyGen[1, Random[Integer,{0,1}]]},
                        {Q, PolynomialQ, polyGen[Random[Integer,{2,5},
                                                Random[Integer,{0,1}]}
                }],
                resolutionActions[{
                        accion1,
                        accion2,
                        ... ,
                        accionR}
]
]
```

Figura A.3. Ejemplo de definición de una nueva estrategia correspondiente a un tipo predefinido

D) DEFINICIÓN DE ACCIONES DE UNA ESTRATEGIA

La estructura mostrada en la figura siguiente muestra resaltada la cabecera de *resolución de acciones* (*resolutionActions*). Esta cabecera identifica la lista de acciones que va a introducir el diseñador para la resolución del ejercicio por parte del alumno o para la resolución guiada por parte de *MathTrainer*. Este proceso se describe con detalle en el apartado 3.3.3.2.2 de la memoria.

```
id_tipo_de_ejercicio[
        enunciado,
        id_estrategia[
                etiqueta,
                descripción,
                formulaePatterns[Rule[id_formula,pattern]__],
                cases[{metaVarsSpec[__metaVariables]__}],
                resolutionActions[{
                        accion1,
                        accion2,
                        ... ,
                        accionN}]
]
```

Figura A.4. Cabecera para la lista de acciones de resolución.

E) ESTRUCTURA DE DATOS RESULTANTE TRAS LA DEFINICIÓN DE UNA ACCIÓN DE *selección de alternativa*.

En el apartado 3.3.3.2.2 se describe con detalle el proceso necesario para la definición de una colección de alternativas para ser ofrecidas al alumno en tiempo de resolución en una paleta de selección excluyente. Un ejemplo de la estructura de datos resultante tras el proceso de definición de los datos relativos a la acción *selección de alternativa* es el que se muestra en la figura A.5.

Selección de alternativa[{

{"u·v − ∫(u·v)dx ", mensaje["Incorrecto. Repasa la teoría de integración por partes"]},

{"u·V + ∫(V·du)dx ", mensaje["Incorrecto. Repasa la teoría de integración por partes"]},

{"u·V − ∫(V·du)dx ", entrada de expresiones["Introduce la derivada de u", du, D[u,x]], asignar[nuevoEjercicio, Integrar[Vdu,x]]}

}]

Figura A.5. Ejemplo de alternativas de la acción *selección de alternativa*.

Hemos resaltado en negrita las acciones que el motor de resolución de *MathEdu Solver* invocará en tiempo de ejecución. La acción ***selección de alternativa*** muestra la paleta y cada una de las otras acciones (*mensaje* y *entrada de expresiones*) se ejecutarán según el alumno seleccione los botones con la etiqueta mostrada en los campos de descripción.

F) DEFINICIÓN DE UN SUBPROBLEMA MEDIANTE LA ACCIÓN *resolución de ejercicio.*

Desde el punto de vista de la estructura de datos, lo que sucede al diseñar una acción de tipo *resolución de ejercicio* se muestra en la figura A.6 a continuación. La interfaz está descrita con detalle en la sección 3.3.3.2.2. La idea básica es que hay que relacionar elementos que aparecen descritos en dos ejercicios, uno ya ha sido definido previamente y forma parte del curso y otro, el ejercicio nuevo, el cual está en proceso de definición y que se debe apoyar en el primero para resolver una parte concreta del mismo.

En el ejemplo concreto que presentamos aquí el tipo de ejercicio predefinido es adición. Es un tipo de ejercicio que nos permite calcular la suma de dos cifras. Puesto que en el ejercicio nuevo que estamos definiendo aparece una suma de dos valores, parece apropiado usar el tipo de ejercicio adición para resolver ese subproblema. Para ello hemos de obtener el resultado de la suma en el primer factor de la expresión a resolver. Por este motivo invocamos una acción de *resolver ejercicio* en la cual los datos de entrada al subproblema son *n1* y *n2*, y el resultado de su suma se almacenará en la variable *resultadoSuma*. Puesto que *factor1* representa la variable que contendrá la salida del subproblema y se define como *resultadoSuma*, una vez que concluya el subproblema podremos referirnos a la variable *factor1* a sabiendas de que en ella estará contenido el valor de la suma buscada.

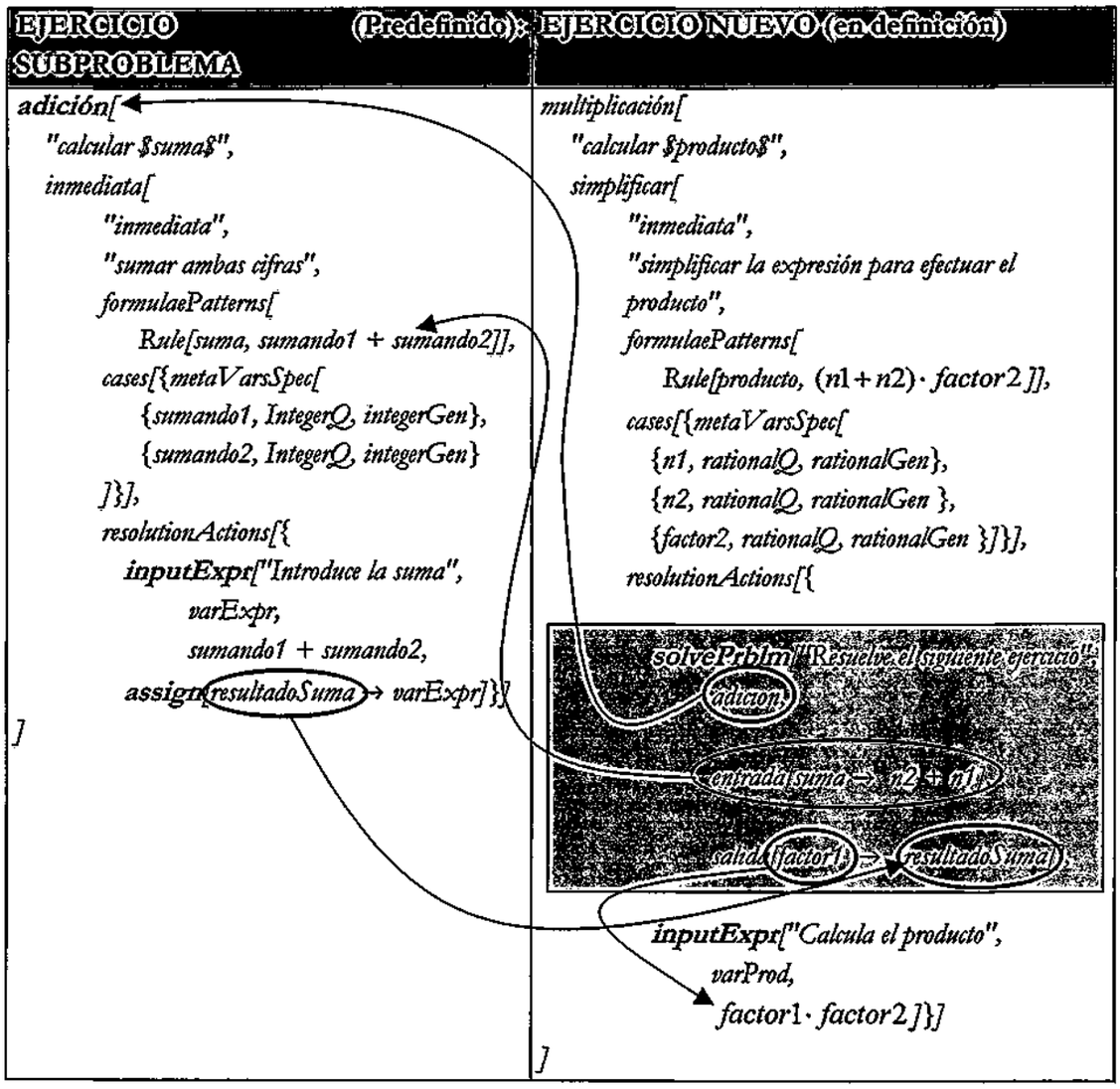

Figura A.6. Interacción de datos en el diseño de la acción resolución de ejercicio.

G) GENERACIÓN ALEATORIA DE EJERCICIOS

En la figura A.7 que mostramos a continuación se describe el seudocódigo correspondiente a la función encargada de la generación aleatoria de ejercicios, proceso descrito en el apartado 3.4.1 de la memoria. Como se puede apreciar, el proceso consiste en

- Leer los tipos de ejercicio del curso
- Escoger aleatoriamente un tipo de entre todos los posibles
- Escoger aleatoriamente una estrategia entre las asociadas al tipo seleccionado
- Invocar la resolución del ejercicio

```
seleccionAleatoriaDeEjercicio :=

        EXERCISES = lee(curso.txt);

        numeroTipos = contarTiposDeEjercicios(EXERCISES);

        tipoSeleccionado = Random(1, numeroTipos);

        numeroEstrategias = contarEstrategias(tipoSeleccionado);

        estrategiaSeleccionada = Random(1, numeroEstrategias);

        solveProblem(
                "Enunciado del ejercicio",
                tipoSeleccionado,
                entrada(),
                salida());
```

Figura A.7. Seudocódigo de la función de generación aleatoria de ejercicios.

La función **solveProblem** la cual posee cuatro parámetros:

- Mensaje para el cuaderno de diseño
- El tipo de ejercicio seleccionado
- Entrada()
- Salida()

Como podemos observar, el tercer parámetro corresponde a la cabecera entrada( ), la cual no lleva parámetros. De esta forma la función **solveProblem** identifica que el ejercicio es uno nuevo e invoca a la función de evaluación de datos de entrada para el ejercicio (**evaluarEntrada**) la cual genera los valores correspondientes para las metavariables que aparecen definidas en el enunciado.

La función **evaluarEntrada** accede a las metavariables correspondientes a la estrategia escogida del tipo de ejercicio seleccionado, toma sus funciones generadoras y genera los valores correspondientes. Dichos valores se sustituyen en los identificadores de las metavariables y se aplican las reglas correspondientes de los patrones de fórmula con los que se inicializan los identificadores de las mismas. Una vez concluido este proceso ya se puede enviar el enunciado al cuaderno de resolución, sustituyendo en el mismo los identificadores de las fórmulas por los valores generados.

En la figura siguiente queremos mostrar como es el proceso descrito hasta el momento correspondiente a la generación aleatoria de ejercicios.

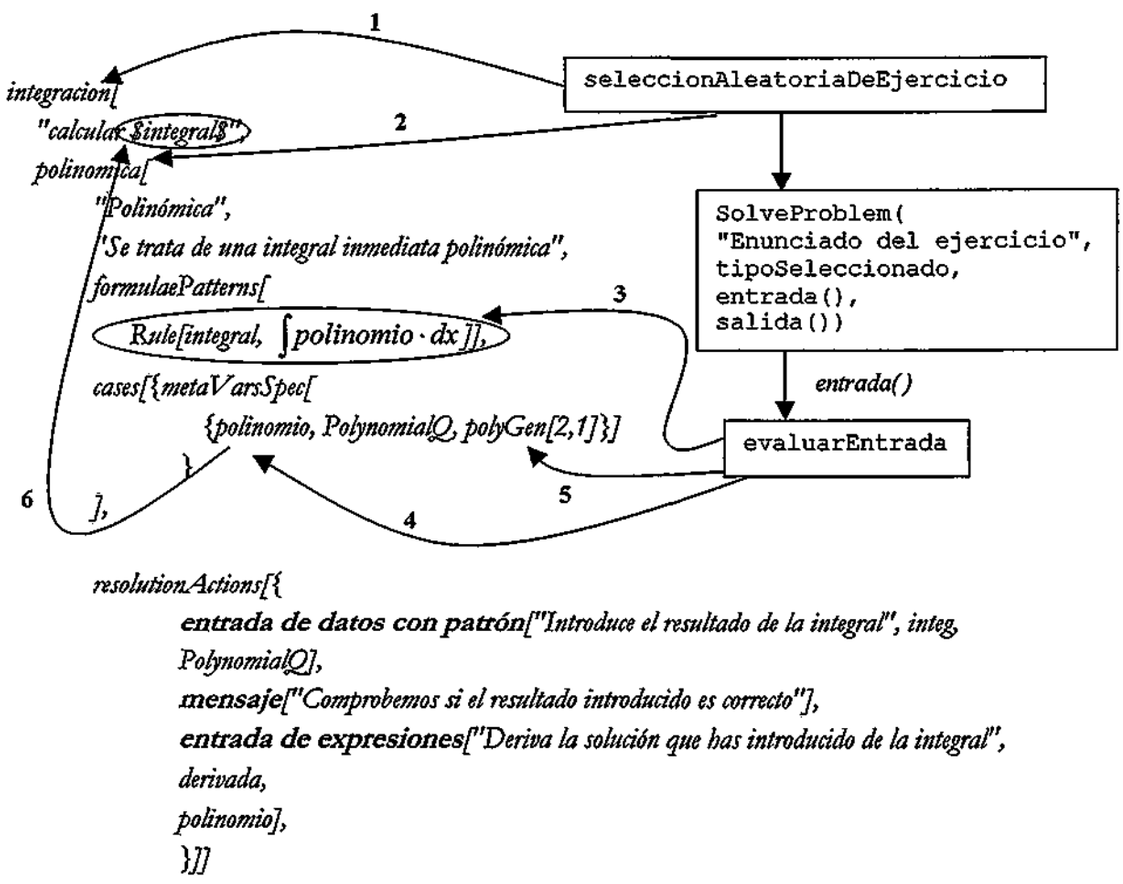

Figura A.8. Representación gráfica de la generación aleatoria de un ejercicio.

Los pasos 1 y 2 corresponden a la función **seleccionAleatoriaDeEjercicio**, la cual se encarga de la selección del tipo de ejercicio (1) y de la estrategia correspondiente al mismo (2). Esta función invoca a su vez a **solveProblem** la cual se encarga de analizar el tipo de entrada de datos e invoca, a su vez, a la función de evaluación de datos de entrada **evaluarEntrada**. Esta última función lleva el peso, a su vez, de obtener los patrones de las fórmulas asociadas a la estrategia seleccionada (paso 3), obtener las metavariables de las fórmulas involucradas en los patrones anteriores (paso 4), generar los valores de las metavariables con sus correspondientes funciones generadoras (paso 5) y propagar (paso 6) los valores de las fórmulas con los valores de las metavariables instanciados al enunciado del ejercicio que se muestra en el cuaderno de resolución.

H) VERIFICACIÓN DE PATRONES

La figura A.9 que mostramos a continuación reproduce el seudocódigo del procedimiento de verificación de patrones de *MathEdu Solver*. Los pasos asociados a esta función están descritos en la sección 3.4.2.

```
verificaciónDePatrones(tipo_, etiqueta_, expresion_):=

(* 1. OBTENCIÓN DE DATOS *)
        ejercicio = leeEjercicio(tipo);

        estrategia = leeEstrategia(ejercicio, etiqueta);

        metaVars = leeMetaVariables(estrategia);

        liberar(metaVars);

(* 2. GENERACIÓN DE PATRONES *)
        patron = seleccionarPatronFormula(estrategia);

        patronCondicional = patternGenerator(patron, metaVars);

(* 3. DECISIÓN *)
        decision = comparacionSemantica(expresion, patronCondicional);

(* 4. RECUPERACIÓN *)
        recuperacionValoresMetaVariables;

(* 5. SALIDA *)
        Switch(decision)
        True -> Return(True)
        False -> errorMessage("estrategia seleccionada incorrecta")
```

Figura A.9. Seudocódigo de la función de verificación de patrones de expresiones.

I) ESTRATEGIAS MÚLTIPLES

La figura A.10 que mostramos a continuación reproduce la estructuras de datos correspondiente a un ejercicio (en este caso de tipo *multiplicación*) que contempla la posibilidad del uso de dos estrategias diferentes para su resolución. En la sección 3.4.2 se detalla el uso de este tipo de estructuras.

multiplicacion[
enunciado,
distributiva[descripción,
patrones de fórmulas,
casos,
acciones[accion_d1,accion_d2,... ,accion_dK],
simplificar[descripción,
patrones de fórmulas,
casos,
acciones[accion_s1,accion_s2,... ,accion_sT]
]

Figura A.10. Ejemplo de estructura de datos de un ejercicio con dos estrategias.

J) BUCLE DE EJECUCIÓN DE ACCIONES

La figura A.11 que mostramos a continuación reproduce el seudocódigo de la función **hacerAcciones**, la cual aparece referida en la sección 3.4.3.

```
hacerAcciones({acciones_}):=

ejecutarAccion = extraerPrimera(acciones);

Switch(ejecutarAccion,

        entradaDatos(__) → inputData(__, Rest(acciones));
        entradaExpresiones(__) → inputExpression(__, Rest(acciones));
        entradaDatosCondicional(__) → inputConditional(__, Rest(acciones));
        mensaje(__) → showMessage(__, Rest(acciones));
        asignar(__) → assignVariable(__, Rest(acciones));
        seleccionAlternativa(__)→choicePaletteIteration(__, Rest(acciones));
        resolverProblema(__) → solveExercise(__, Rest(acciones));
        finSubproblema(__) → endSubproblem(__, Rest(acciones));
        finResolución(__) → endResolutionExercise(__, Rest(acciones));
)
```

Figura A.11. Seudocódigo de la función que realiza el bucle de acciones.

La potencia del *patter-matching* de *Mathematica* permite que la función tenga un aspecto tan simple como el que se aprecia en la figura. Si observamos cada una de las acciones que aparecen en la primera columna sólo llevan como parámetros un doble símbolo de subrayado __. *Mathematica* interpreta que dicho símbolo es equiparable a la presencia de uno o más parámetros asociados a la cabecera de la función. Por tanto lo realmente relevante son las cabeceras, independientemente del número de parámetros que porten.

Por otro lado también conviene aclarar que la *traducción* de las funciones que representan los tipos de acciones a efectuar por el alumno a funciones ejecutables por el núcleo de *Mathematica* se realiza aplicando la traducción a la correspondiente función ejecutable de cada tipo de acción a los parámetros que porta la declaración de dicha acción. Así, por ejemplo, la acción

entrada de expresiones("Efectúa el primer producto", varProd1, factorsumando1)

se transforma, aplicando la función ejecutable, en

inputExpression(*"Efectúa el primer producto", varProd1, factorsumando1, resto_acciones*)

que como podemos observar introduce la lista de acciones restantes como último parámetro, con el fin de mantener el bucle de acciones.

K) EVALUACIÓN DE CONDICIONES SOBRE EXPRESIONES

La comprobación de la condición la efectúa una función de evaluación de predicados la cual debe distinguir el condicionante para efectuar la comprobación oportuna. Su seudocódigo es:

```
evaluarPredicado(expresión, condición):=

    Si condición es de la forma "*Q"
        entonces hacer
        condición(expresión);
    en caso contrario hacer
        expresión === condición
```

Su interpretación es muy simple. Se basa en el hecho de distinguir el tipo de condición indicada. Si la condición es cualquier cadena de caracteres terminada en Q (que indica *Question*, es un predicado), se aplica directamente el predicado a la expresión que teclee el alumno. Por ejemplo

$$\text{racionalQ}(\frac{1}{x}) \quad \Rightarrow \quad \text{Verdadero}$$

mientras que si la condición no es de la forma anterior debe ser la expresión correspondiente a un patrón semántico, en cuyo caso la comparación con la expresión tecleada por el alumno se hace a través de la triple igualdad (= = =) de *Mathematica*. Dicha triple igualdad equivale, a su vez, al predicado **SameQ** el cual compara las expresiones a ambos lados y comprueba si son idénticas o no. Por ejemplo

$$\frac{Sen(2x+2)}{Cos(2x+2)} === Tg(2x+2) \qquad \Rightarrow \qquad \text{Verdadero}$$

comprobando de este modo que el predicado **SameQ** es capaz de resolver simplificaciones sencillas como las del ejemplo. Una cuestión pendiente de estudio respecto al funcionamiento de este predicado en *Mathematica* es su alcance. Es decir, hasta qué punto es capaz de realizar simplificaciones o, dicho de otro modo, cuándo es necesario incluir en la función evaluarPredicado funciones de simplificación propias de *Mathematica* como **Simplify** o **FullSimplify**, funciones ambas las cuales de momento no usamos. Esta, que duda cabe, debe ser una cuestión para tratar en futuras versiones de la aplicación y, con

toda seguridad, será pertinente acudir a las fuentes de *Mathematica* (Wolfram Research) para obtener información adicional no incluida en los manuales del producto.

APÉNDICE B

EXPRESIONES SIMBÓLICAS EN *MATHEMATICA*

Mathematica manipula dos tipos de expresiones simbólicas. Por un lado las puramente matemáticas. Por otro lado las expresiones gráficas. Para construir las segundas, las cuales describen el contenido de los *notebooks*, *Mathematica* dispone de un lenguaje capaz de interpretarse a modo de guión. Por ejemplo, consideremos la celda siguiente

```
1 + 1
```

cuya representación en el lenguaje de *Mathematica* es

```
Plus[1,1]
```

y cuya representación en el lenguaje de cajas que *Mathematica* emplea para representar expresiones en el *front-end* es

```
Cell[BoxData[RowBox[{"1", "+", "1"}]], "Input"]
```

Este código se puede ver y editar de forma interactiva en el *front end.* De hecho *MathEdu* manipula este tipo de expresiones directamente cuando es necesario hacerlo para enviarlas a la interfaz con el usuario (alumno o profesor). Todas las celdas en *Mathematica* se obtienen de esta forma. El lenguaje de *boxes* (cajas) es altamente versátil puesto que debe ser capaz de representar cualquier expresión matemática que se desee. Existen numerosas primitivas en el lenguaje que permiten crear cualquier tipo de expresión, con cualquier color, tamaño, fuente, etc.

Semejantes capacidades nos han provocado, a menudo, evidentes dificultades en la manipulación de los datos. La interfaz con el usuario sólo es capaz de manipular la información cuando está en el lenguaje de cajas. Por este motivo para mostrar cualquier expresión siempre hemos de traducirla entre los distintos formatos usando para ello fundamentalmente dos funciones: **ToExpression** y **ToBoxes**. Para efectuar estas traducciones *Mathematica* dispone de funciones primitivas que permiten el paso de expresiones del formato simbólico gráfico al simbólico matemático y viceversa (ver figura B.1).

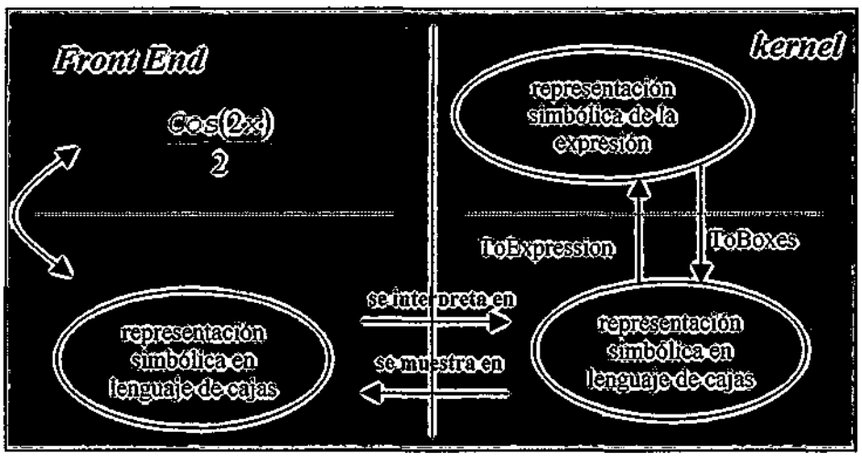

Figura B.1. Esquema de las representaciones simbólicas en *Mathematica*.

A modo de ejemplo vamos a mostrar una manipulación de datos habitual en *MathEdu*. El alumno teclea la expresión

$$\frac{\cos(2x)}{2}$$

cuya representación en lenguaje de cajas, representación interna de la expresión que manipula el *front end*, es

```
Cell[BoxData[
        FractionBox[
                RowBox[{"Cos", "[",
                RowBox[{"2", "x"}], "]"}], "2"]], "Input"]
```

Esta expresión contiene información gráfica relativa a la disposición de los distintos elementos en la celda, pero desde el punto de vista matemático es totalmente inoperativa. Para poder manipular su representación matemática hemos de crear una copia de esta expresión en el *kernel*, que es donde realmente se interpreta. Una vez en el *kernel*, disponemos de una forma de transformarla en una expresión de contenido matemático pleno mediante la primitiva **ToExpression**, convirtiéndose en

```
Times[Rational[1,2], Cos[Times[2,x]]].
```

Supongamos ahora que dicha expresión la multiplicamos por 2, obteniendo

$$\cos(2x)$$

Hemos de transformarla nuevamente a formato *boxes* para podérsela mostrar al alumno. Para ello disponemos nuevamente de otra primitiva, en este caso **ToBoxes** de modo que

```
ToBoxes[Cos[Times[2,x]]]
Cell[BoxData[
        RowBox[{"Cos", "[",
        RowBox[{"2", "x"}], "]"}]], "Input"]
```

El *kernel* transfiere una copia de esta última expresión simbólica en lenguaje de cajas al *front end*, el cual la muestra en el *notebook* correspondiente.

En resumen, las expresiones siempre se interpretan, en cualquiera de sus formatos, en el *kernel*. El *front end* actúa como un mero procesador de textos que muestra la información que se introduce desde el teclado o se procesa desde el *kernel*.

REFERENCIAS BIBLIOGRÁFICAS

Referencias Bibliográficas

- Allen, R. (1997). Informe del Grupo de Trabajo sobre **Entornos Interactivos de Aprendizaje** en Actas del 8° Congreso Internacional de Educación Matemática. Sevilla, España, pp. 353-357.

- Anderson, J.R. (1983). **The Architecture of Cognition**. Harvard University Press, Cambridge, Massachusets.

- Anderson, J.R.; Boyle, C.F.; Corbett, A.T.; Lewis, M.W. (1990). **Cognitive Modeling and Intelligent Tutoring**, en Artificial Intelligence 42.

- Bauer, M.; Dengler, D.; Paul, G.; Meyer, M. (2000). **Programming by demonstration for Information Agents** en Communications of the ACM, vol 43, No. 3.

- Bazin, J.M., Castells, P., Moriyón, R. y Saiz, F. (1993). **A Knowledge Based Problem Solver Conceived for Intelligent Tutoring Applications**, Proceedings ICCTE'93, Kiev.

- Bellemain, F. (1992). **Conception, réalization et expérimentation d'un logiciel d'aide à l'enseignement de la géometrie: Cabri-géomètre**. Tesis Doctoral. Grenoble, France: Université Joseph Fourier.

- Bernat, P. (1996). **CHYPRE: an interactive environment for elementary geometry problem solving.** Accesible en: http://www-didactique.imag.fr/preuve/Resumes/Bernat/Chypre/Chypre.html

- Biegel, J.E. (1989). **The essential components of an intelligent simulation training system** en Simulation and Artificial Intelligence. Society for Computer Simulation International.

- Bronte, R. (1974). **Problemas de cálculo infinitesimal e integral.** Litoprint.

- Capani, A; (2000). **The Design of CoCoA 3 System**, Tesis Doctoral. Departamento de Informática y Ciencias de la Información. Universidad de Génova.

- Carro, R.; Pulido, E.; Rodríguez, P. (1999a) **An adaptive driving course based on HTML dynamic generation**. *Top Paper Award* en la World Conference on the WWW and Internet, WebNet'99. Hawai, USA.

- Carro, R.; Pulido, E.; Rodríguez, P. (1999b). **Designing Adaptive Web-based Courses with TANGOW**, en Cumming, G., Okamoto, T., Gomez, L. *Advanced Research in Computers and Communications in Education*, (2). Amsterdam, IOS Press.

- Castells, P., Moriyón, R. y Saiz, F.; Villa, E. (1993). **Applications of Techniques of Automatic Problem Solving to Computer Intelligent Tutoring,** Actas de ICCTE'93, Kiev.

- Castells, P., Moriyón, R. y Saiz, F.; Villa, E. (1993b). **A Model of Knowledge in Elementary Mechanics**, Actas de ICCE'93, Taiwan.

- Castells, P.; Moriyón, R.; Saiz, F. (1995). **Solving Mathematical Problems that Involve Reasoning and Calculations**. Golden West International Conference on Intelligent Systems. ACM. San Francisco.

- Cypher, A. (1993). **Watch what I do. Programming by Demonstration.** Cambridge, MA. The MIT Press.

- Dieudonné, J. (1968). **Calcul infinitesimal.** Hermann, París.

- Diego Martín, B. (1987). **Ejercicios de análisis cálculo diferencial e integral.** Deimos, Madrid.

- Díez, F.; García, M.C. (1996). **Supervisión automática del aprendizaje matemático,** Póster presentado en el *8º Congreso Internacional de Educación Matemática (ICME-8).* Sevilla (España).

- Díez, F.; García, M.C. (1997). ***AMNESIA*: Un supervisor automático del aprendizaje matemático**, Actas del *Congreso Internacional de Informática Educativa*, UNED. Madrid, (España).

- Díez, F.; Moriyón, R. (1999). **Doing Mathematics with MathEdu**, Actas de la IXth Conference of Mathematics/Science Education & Technology. AACE, San Antonio, (EE.UU.).

- Díez, F.; Moriyón, R. (2000). **Symbolic Calculus Training by Means of *MathTrainer.*** En actas del 2º Symposio Internacional de Informática Educativa. ADIE, Puertollano, (España).

- Díez, F.; Moriyón, R. (2001). **Teaching Mathematics by Means of MathTrainer,** En actas de la 12th International Conference of the Society for Information Technology & Teacher Education. AACE, Orlando, (EE.UU.).

- Gallego, A.; Martínez, E. (2000). **Análisis de una experiencia de enseñanza virtual en la Universidad politécnica de Cartagena,** póster presentado en el 2° Simposium Internacional de Informática Educativa. Puertollano, España.

- Gertner, A.S.; VanLehn, K. (2000). **Andes: A Coached Problem Solving Environment for Physics,** Actas de la 5ª Conferencia Internacional ITS 2000. Montreal, (Canadá).

- Glynn, J.; Gray, T. (1997). **The Beginner's Guide to *Mathematica* version 3.** Cambridge University Press.

- Greg, J. (1998). **The LEIBNIZ Front End.** Actas de la Worldwide Mathematica Conference. Chicago.

- Gutiérrez, J. (1994). **INTZA: Un Sistema Tutor Inteligente para Entrenamiento en Entornos Industriales.** Tesis Doctoral. Facultad de Informática, Universidad del País Vasco.

- Halbert, D. (1981). **An Example of Programming by Example.** Tesis Doctoral. University of California Berkeley.

- Hiemstra, R.; Brockett, R. G. (1994). **From behaviorism to humanism: Incorporating self-direction in learning concepts into the instructional design process.** En H.B. Long & Associates (Eds.) New ideas about self-directed learning. Norman, OK University of Oklahoma.

- Ifrah, G. (1998). **Historia Universal de las cifras.** Espasa-Calpe, Madrid.

- Jackiw, R. (1992). **The Geometer's Sketchpad.** Key Curriculum Press, Berkeley, CA.

- Johassen, D. H. (1991). **Evaluating constructivist learning.** Educational Technology, 31(9).

- Kay, J.; Kummerfeld, R.J. (1994). **An individualized course for the C programming language.** Actas de la Second International WWW Conference, Chicago, IL. Accesible en http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Educ/kummerfeld/kummerfeld.html

- Kearsley, G.P. (1983). **Computer based training: A guide to the selection and implementation**. Reading, M.A. Eds. Addinson-Wesley.

- Koedinger, K.R.; Anderson, J.R.; Hadley, W.H.; Mark, M.A. (1997). **Intelligent Tutoring goes to the school in the big city**. International Journal of Artificial Intelligence in Education, 8(1).

- Koffman, E.B.; Blount, S.E. (1975). **Artificial Intelligence and automatic programming in CAI**. En Articial Intelligence. Vol. 6.

- Laird, J.; Bates, C.; Altmann, E.; Doorembos, R. (1993). **Soar user's manual**. The Soar Group. School of Computer Science, Carnegie Mellon University. Pittsburgh PA.

- Lieberman, H.; Hewitt, C. (1980). **A Session with Tinker: Interleaving Program Testing With Program Design**. Conference Record of the 1980 LISP Conference, Stanford University.

- Lieberman, H (2000). **Programming by Example**. Communications of the ACM, vol 43, No. 3.

- Lorenzo, J. (1989). **El estilo matemático**. Ed. Tecnos, Madrid (España).

- Masthoff, J.F.M. (1997). **An agent-based interactive instruction system**. Tesis Doctoral. University of Technology Eindhoven (Holanda).

- Mei-Chuen Lin, J. (1999). **An Internet-Based CAL Software for Solving Trigonometric Problems**. Actas de la IXth Conference of Mathematics/Science Education & Technology. AACE. San Antonio (EE.UU.).

- Myers, B. (1988). **Creating User Interfaces by Demonstration**. Academic Press, San Diego.

- Myers, B; Rosson, M. (1992). **Survey on User Interface Programming**. Proceedings of CHI'92, ACM, Monterey, CA.

- Myers, B.A.; McDaniel, R.G.; Miller, R.C.; Ferrency, A.S.; Faulring, A.; Kyle, B.D.; Mickish, A.; Klimovitski, A.; Doane, P. (1997). **The Amulet Environment: New Models for Effective User Interface Software Development**. IEEE Transactions on Software Engineering, Vol. 23, no. 6.

- Myers, B; Rosson, M. (2000). **Intelligence in Demonstrational interfaces.** en Communications of the ACM, vol 43, No. 3.

- Morrisroe, G.C. (1986). **An application of Intelligent Tutoring Systems to Air Traffic Control Training.** En IEEE Colloquium on Intelligent Tutorial Systems. Londres (Reino Unido).

- Newell, A. (1990). **Unified Theories of Cognition.** Cambridge, Mass. London: Harvard University Press.

- Clancey, W.; Letsinger, R. (1981). **NEOMYCIN: reconfiguring a rule-based expert system for application to teaching.** En Actas de la the 7th International Joint Conference on Artificial Intelligence. Vancouver, Canada.

- O'Shea, T.; Self, J. (1985). **Enseñanza y aprendizaje con ordenadores.** Ed. Anaya Multimedia S.A.

- Pitrat, J. (1990). **Métacoinnaisance, futur de l'intelligence artificielle.** Hermes, Paris.

- Puerta, A.R. (1998). **Supporting User-Centered Design of Adaptive User Interfaces Via Interface Models.** Actas del First Annual Workshop On Real-Time Intelligent User Interfaces For Decision Support And Information Visualization, San Francisco.

- R.A.E.C. (1971). **Problemas de cálculo integral.** Ediciones Universitarias y Cultura, Madrid.

- Ritter, F.; Feurzig, W. (1988). **Teaching Real Time Tactical Thinking,** en Intelligent Tutoring Systems: lessons learned. Ed. Psotka, J.; Massey, L.D.; Mutter, S.A.

- Rosenbloom; Laird; Newell (1993). **The Soar Papers: Research on Integrated Intelligence.** Cambridge. MIT Press.

- Self, J. (1974). **Student models in computer-aided instruction.** International Journal of Man Machine Studies. Academic Press, Cambridge (Reino Unido).

- Serna, A.; Cordero, N.A.; Tricio, V.; Ballesteros, A. (1999). **Estudio interactivo con *Mathematica* del límite continuo de un péndulo múltiple.** Actas del primer Congreso Nacional de Informática Educativa.Puertollano, España.

- Skinner, B. F. (1938). **The behavior of organisms.** Appleton-Crofts, Nueva York.

- Sleeman, D. (1982). **Assessing aspects of competence in basic algebra.** En Intelligent Tutoring Systems. Academic Press, New York.

- Smith, D. (1975a). **Pygmalion: A Computer Program to Model and Stimulate Creative Thought.** Birkhauser, Basel.

- Smith, D. (1975b). **Pygmalion: A Creative Programming Environment.** Informe número STAN-CS-75-499. Department of Computer Science, Stanford University.

- Spivak, M. (1980). **Calculus.** Ed. Reverté, Barcelona.

- Sutcliffe, G. y Suttner, C.B. (1997). **The CADE-13 ATP System Competition.** Journal of Automated Reasoning, 18(2).

- Szekely, P., Sukaviriya, P., Castells, P., Muthukumarasamy, J. y Salcher, E. (1996). **Declarative Interface Models for User Interface Construction Tools: the MASTERMIND Approach.** Engineering for Human-Computer Interaction. Eds. L. Bass y C. Unger. Chapman & Hall.

- Wenger, E. (1987). **Artificial Intelligence and tutoring Systems: computational approaches to the communication of knowledge.** Los Altos, California. Morgan Kauffmann Publishers, Inc.

- Whitaker, E.T.; Bonnell, R.D. (1992). **A Blackboard Model for Adaptive and Self-Improving Intelligent Tutoring Systems.** En Journal of Artificial Intelligence in Education (1992) 3.

- White, B.Y.; Frederiksen, J.R. (1986). **Intelligent Tutoring Systems Based upon Qualitative Modell Evolutions,** en Actas de la 5th National Conference on Artificial Intelligence. Philadelphia. Pennsylvania.

- Wolfram, S. (1991). **Mathematica, A System for Doing Mathematics by Computer.** Wolfram Research, Inc.

- Wolfram, S. (1999): **The Mathematica Book.** Cambridge University Press.

Reunido el tribunal que suscribe en el día
de la fecha, acordó calificar la presente Tesis
doctoral con SOBRESALIENTE CUM LAUDE POR UNANIMIDAD
Madrid, 6-3-2002

Francisco Saiz

Isabel Fernandez - Luque de Jorge

Manuel Alfonseca

Ignacio Aedo